STEM-Sat

Using Cubesats in the pre-K to 12 STEM Curricula

A Resource Guide for Educators

Number 1 in the STEM Series

Patrick H. Stakem

Table of Contents

"[Science] is more than a school subject, or the periodic table, or the properties of waves. It is an approach to the world, a critical way to understand and explore and engage with the world, and then have the capacity to change that world..."

President Barack Obama, March 23, 2015

Introduction

This book is targeted to STEM (Science, Technology, Engineering, and Mathematics) educators and administrators. It is intended to show how Cubesats in particular, and Space topics in general can be applied to their STEM program. It is a train-the-trainer document, pointing out sources of information and support to the teaching professional who knows how to present the material at the appropriate level.

The approach for space-related projects for STEM allows small working groups to focus on projects of their own choice and interest. This approach works because just about any interest can be accommodated in a space mission.

It is anticipated that most Cubesat projects would not be focused on flight, but it is a possibility. The first K-8 Cubesat to be launched was from St. Thomas More School in Arlington, Virginia in July, 2016. It was named CathedralSat-1, and went to the International Space Station for deployment into orbit. It is still returning image data as of this writing. It will involve multi-semesters, with the data and project being handed down from one student group to another. This always drives home the value of good documentation.

It is not impossible for a STEM project to get a flight opportunity. The launch costs are fairly large, but NASA itself and many aerospace corporations fund the launch of worthy projects.

In addition, testing of Cubesat payloads can be done by medium and high altitude balloon, and by model rocketry. This book will

also provide the educator pointers to the vast amount of applicable material that NASA has available and is freely given.

From the material in this book, a pre-K-to-12 educator should be able to develop age- and grade-appropriate presentation material. I have no clue how to do that – my teaching experience has been with highly motivated senior undergraduate and graduate students. But, I do know the material.

This will be an evolving document, as I can capture more "best practises" from existing STEM Cubesat Programs. This book will have a companion website for your feedback and questions, and for you to collaborate with similar projects.

Author

Mr. Patrick H. Stakem has been fascinated by the space program since the Vanguard launches in 1957. He received a Bachelors degree in Electrical Engineering from Carnegie-Mellon University, and Masters Degrees in Physics and Computer Science from the Johns Hopkins University. At Carnegie, he worked with a group of undergraduate students to re-assemble, modify, and operate a surplus missile guidance computer, which was later donated to the Smithsonian. He was brought up in the mainframe era, and was taught to never trust a computer you could lift.

He began his career in Aerospace with Fairchild Industries on the ATS-6 (Applications Technology Satellite-6) program, a communication satellite that developed much of the technology for the TDRSS (Tracking and Data Relay Satellite System). He followed the ATS-6 Program through its operational phase, and worked on other projects at NASA's Goddard Space Flight Center including the Hubble Space Telescope, the International Ultraviolet Explorer (IUE), the Solar Maximum Mission (SMM), some of the Landsat missions, and Shuttle. He was posted to NASA's Jet Propulsion Laboratory for Mars-Jupiter-Saturn (MJS-77), which later became the *Voyager* mission, and is still operating and returning data from outside the solar system at this writing. He

initiated and lead the international Flight Linux Project for NASA's Earth Sciences Technology Office. He is the recipient of the Shuttle Program Manager's Commendation Award, and has completed 42 NASA Certification courses. He has two NASA Group Achievement Awards, and the Apollo-Soyuz Test Program Award.

Mr. Stakem has been affiliated with the Whiting School of Engineering of the Johns Hopkins University since 2007, and Capitol Technology University. Mr. Stakem supported the Summer Engineering Bootcamp Projects at Goddard Space Fight Center for 2 years. This resulted in the Greenland Rover project being deployed. That is essentially a zero-altitude satellite that measures the thickness of the Greenland Ice Sheet. He developed and presented Cubesat courses at the undergraduate and graduate level for several years. He continues to work on Cubesat Projects with his past students collaboratively world-wide.

What is a Cubesat?

A Cubesat is a small, affordable satellite that can be developed and launched by colleges, high schools, and even individuals. The specifications were developed by Academia in 1999. The basic structure is a 10 centimeter cube, (volume of 1 liter) weighing less than 1.33 kilograms. This allows multiples of these standardized packages to be launched as secondary payloads on other missions. A Cubesat dispenser has been developed, the Poly-PicoSat Orbital Deployer, (P-POD) that holds multiple Cubesats and dispenses them on orbit. They can also be launched from the Space Station, via a custom airlock. ESA, the United States, Russia, and others provide launch services. The Cubesat origin lies with Professor Twiggs of Stanford University and was proposed as an approach to support hands-on university-level space education and opportunities for low-cost space access. This was defined at a presentation at the University Space Systems Symposium in Hawaii in November of 1999.

Cubesats began as teaching tools, and remain in that role, although

their vast numbers in orbit shows that they have become mainstream.

In what has been called the Revolution of smallsats, Cubesats lead the way. They represent paradigm shifts in developing space missions, opening the field from Nationally-funded efforts and large Aerospace contractors, to individuals and schools.

Even if your personal Cubesat project never gets launched or even built, it will bring you valuable experience to participate. This book will introduce and explain the NASA Systems Engineering Process, which leads from a set of goals to a successful space flight. This model has been in use for decades, and has proven itself to have usefulness. It has been refined as problems were uncovered, and still remains a viable approach to space missions, as well as regular engineering applications.

Cubesats can be custom made, but a major industry has evolved to supply components, including the structure, electronics, batteries, and computers. It allows for an off-the-shelf implementation, in addition to the custom build. There is quite a bit of synergy between the Amateur Satellite (Amsat) folks and Cubesats. NASA supports the Cubesat program, holding design contests and providing free launch services to worthy projects. Cubesats are being developed around the world, and several hundred have been launched.

Build costs can be lower than $10,000, with launch costs ranging around $100,000, a most cost-effective price for achieving orbit. The low orbits of the Cubesats insure eventual reentry into the atmosphere, so they do not contribute to the orbital debris problem.

Central to the Cubesat concept is the standardization of the interface between the launch vehicle and the spacecraft, which allows developers to pool together for launch and so reduce costs and increase opportunities. As a university-led initiative, Cubesat developers have advocated many cost-saving mechanisms, namely:

- A reduction in project management and quality assurance

roles.

- Use of student labor with expert oversight to design, build and test key subsystems.
- Reliance on non-space-rated Commercial-Off-The-Shelf (COTS) components .
- Limited or no built-in redundancy (often compensated for by the parallel development of Cubesats) .
- Access to launch opportunities through standardized launch interfaces.
- Use of amateur radio communication frequency bands and support from amateur ground stations.
- Simplicity in design, architecture and objective .

Multiple cubesats can be carried as secondary payloads on military and commercial flights, Cubesats, as small, inexpensive units have a higher mission risk tolerance. You just have to catch a ride to the orbit you want.

Since the initial proposal of the concept, further efforts have been made to define internal and external interfaces made by various developers of Cubesat subsystems, products, and services that have defined the Cubesat 'standard' as it is today. A core strength of the Cubesat is its recognition of the need for flexibility in the definition of standards, and since conception the standard has evolved to ensure that these design rules are as open as possible. The most significant of these further advances in definition have been for the POD systems (in order to meet launch requirements) and the modularization of the internal electronics.

The in-orbit success rate of university-led Cubesat projects (not withstanding launch failures) is around 50%; this is an understandable result of using the Cubesat as an education tool, where development itself is a learning process. On-orbit failure is a disappointment but should not be considered the primary focus, but

rather a learning experience. For projects involving significant participation of companies with experience in satellite development, all but one were a success and demonstrated the strength of the Cubesat for non-educational applications. A large number of Cubesat missions have demonstrated significant success for on-orbit operations for a sustained period. All Cubesats missions have had technological objectives to some degree, it is the demonstration of devices and system architectures developed in-house, or demonstration of Non-Space-Rated (NSR) Commercial-Off-The-Shelf (COTS) component performance in space.

A simple Cubesat flight controller can be developed from a small standard embedded computing platform such as the Arduino. The lack of radiation hardness is balanced by the short on-orbit lifetime. The main drivers for a Cubesat flight computer are small size, low power consumption, wide functionality, and flexibility. In addition, a wide temperature range is desirable. The architecture can support a real time operating system, but, in the simplest case, a simple loop program with interrupt support will work.

Earth imaging is a common objective for a Cubesat mission, typically achieved using a CMOS camera without any complex lens systems. As it is a critical impediment to the development of a highly capable platform for mission operations, the testing and evaluation of novel approaches for increasing downlink data rate and reliability is also a common objective. While less common than Earth imaging, real science objectives are becoming increasingly popular as recognition (primarily by NASA) of Cubesat capabilities increase and collaborations between engineering and science groups emerge.

Additional capabilities of proposed future missions either in planning or in development include: space weather monitoring, inflatable de-orbit devices, cosmic ray showers, shape memory alloys, star mapping, data relay, re-programmable computing, nano-meteorid dust, plasma probe, and multi-spectral remote

sensing. Solicit the mission from the students. Get them thinking.

Cost reduction in these projects has been achieved through a number of mechanisms, some of which are unavailable to the conventional space industry. The lowest cost yet successful mission is reported to be estimated as under $100,000 including launch costs. A typical cost for a university project varies considerably but a very approximate estimation might be from $50,000 to $150,000 for launch and $5-10,000. in parts cost per unit. Piggyback launches have been offered for free to Cubesats by launch vehicle operators and space agencies.

Another important and related aspect in the design approach is that of modularity in a complete and integrated Cubesat life cycle.

What type of missions can Cubesat's do? Initially, they served as communications relays for Amateur radio. But, they can do essentially what any “big” satellite can do. This includes monitoring space weather, astrophysics, planetary science, or serve as technology demonstrations in orbit.

The Cubesat Design Specification

The Cubesat Design Specification, developed by California Polytechnic State University, defines the physical and interface specifications for Cubesats, and gives testing requirements for vibration, thermal-vacuum tests, and shock, as well as safety. Since a Cubesat flys with other Cubesats in a deployment device, and with an expensive primary payload, safety is a concern. Cubesats are expected to have an on-orbit lifetime of less than 30 years, so as not to clutter up Space.

Here is a synopsis of CubeSat requirements, if you intend to launch your project:

Mass
Each satellite may not exceed 1 kg of mass. The CubeSat center of mass must be within 2 cm of the geometric center.

Structure

All edges contacting rails must be rounded. Cubesats must have at least 75% (85.125 mm of a possible 113.5 mm) of flat rail contact with the deployer.

There are emerging standards for larger Cubesats, multiples of the 1U, such as 6U (12 kg, 12 x 24 x 36 cm), 12U (24 kg, 23 x 24 x 36 cm) and 27U (54 kg, 34 x 35 x 36 cm). These allow the canister to constrain Cubesat deployables such as antennae and solar array. In the original Cubesat specification, this task had to be handled by the Cubesat itself. Even as they get bigger, the standardized architecture and modularity of the Cubesat remains a game-changing advantage. As we will see, the advantages of Cubesats has not been lost on NASA, who has embraced them for low-cost, proof-of-concept missions. Because of this, an entire industry and ecosystem of Cubesat parts and services has developed.

To prevent cold-welding, raw metal is not allowed as the contact surface of the bottom standoff. Derlin inserts, or a hard anodize are examples of acceptable contact surfaces. The outer surfaces of the CubeSats are required to be hard anodized in order to prevent wear between the sliding rails and the CubeSats.

Separation springs (SSMD-51P recommended) must be included at designated contact points. A custom separation system may be used upon approval by CalPoly/Stanford launch personnel.

One deployment switch is required (two are recommended) for each CubeSat.

Material

The use of Aluminum 7075 or 6061-T6 is suggested for the main structure. If other materials are used, the thermal expansion coefficient must be similar to that of Aluminum 7075-T3 (the POD material) and approved by CalPoly/Stanford personnel.

Deployables
A time delay, on the order of several minutes, must be present between release from the P-POD and any satellite hardware deployment, such as solar panels or antennas, to allow for satellite separation. P-POD rails and walls cannot be used to constrain deployable hardware

Communication
There must be a time delay, on the order of several minutes to an hour, before all primary transmitters are activated. Low power beacon transmitters may be activated after deployment. Operators must provide proof of the appropriate license for frequency use.

Power
CubeSats with rechargeable batteries must have the capability to receive a transmitter shutdown command, compliant with FCC regulations. Satellites that require testing and battery charging must provide an external hardware interface to access the power/data port.

A remove-before-flight pin is required to deactivate the CubeSats during integration outside the P-POD. The pin will be removed once the CubeSats are placed inside the P-POD.

General
Absolutely no pyrotechnics are allowed inside the CubeSat. The POD must be able to accommodate satellites with solar panels mounted on the external walls. I hope you and the students get to launch a successful Cubesat mission someday. There's a lot of work and money to achieve that goal. Right now, let's look at this as a learning experience.

What is STEM?

STEM (Science, Technology, Engineering, Mathematics) is the key to the United States' continued dominance in High Technology. It took a lot of expertise to implement the first cell phone. Now they

are turned out like cookies in third world countries.

STEM addresses overall education policy and curriculum sources in schools, at critical grade levels.

Although the teachers are experts in their particular area, and know how to present grade-appropriate material, they may not know how to find and access the resources they need, or where to get help in a particular topic area.

STEM programs are seen as critically important in the education system, world-wide. It is getting to be a complex, interconnected ecosystem. Advances in the subject areas of STEM will take place only if you know how to exploit this ecosystem for knowledge.

When I was in school well before the Internet and STEM age, I had an encyclopedia. Today, students can access WikiPedia from their smart phones. The focus has changed from knowing facts, which are at your fingertips on demand, to leveraging facts to innovate. This approach touches all of the academic disciplines, the Humanities, Languages, Art, besides the STEM topics. Perhaps the best skill set to have is good internet search skills. Teachers have had to transition from asking factual questions, to asking questions that derive from applications of online research, and accrued knowledge.

When I was a kid, there was no STEM. My interests in science and engineering led to research and hands on experimentation. Luckily, I survived. I was called on, while in grade school, to lecture and demonstrate some concepts of electricity to a High School class. The first satellite was launched, and I was glued to the black & white TV. I participated in Model Rocketry at the High School Level, and went on to fly Nationally. This was made possible by an extraordinary High School Science teacher. I made quite a few friends, some of whom became Astronauts. I was given a great opportunity when I received a full scholarship to a College of my choice. I went to Carnegie Tech in Pittsburgh (now, Carnegie-Mellon University), and launched a career in Engineering and Aerospace. It is time for me to pay forward.

I think that Aerospace should be a major focal point for STEM, embracing a wide variety of topics at the cutting edge of technology and science. I have a handful of technical degrees, and spent 42 years at the various NASA Centers. I teach Electrical Engineering courses world-wide, and have done specialty Cubesat courses at the undergraduate and graduate level. It is time to apply that expertise earlier in the education process.

My thesis is, a Cubesat project brings together all of the interesting pieces to provide a focal point for student work. There is a massive body of free support material available from NASA and the Aerospace community. This involves the Education offices at the various NASA Centers, the Visitor's Centers, the speakers' bureaus. I have taken on the task of making STEM educators aware of this vast treasure-trove of resources. I have experience teaching Cubesat engineering and operations at the advanced undergraduate and graduate level, but I have no experience or credentials at the critical pre-K thru 12 levels. Well, I had my grandson build a Cubesat and a solar system model when he was 3. He did a nice job.

I think STEM is a critical resource for understanding and implementing the future. I think the Cubesat paradigm is a good thing to introduce into STEM. Let's do this. Future generations of STEM-mies will thank us. Possibly from the Mars-base.

How do Cubesats fit in?

The Cubesat topic covers a lot of Science, Math, Engineering and Software topics. The Science part encompasses Astronomy and Earth Science, what I call, “Looking Up,” and “Looking Down.”

“Looking Up” is the domain of the Astronomers and Astrophysicists, learning more about the Universe. At closer distances, planetary scientists are exploring the diversity of all the objects in our solar system, and searching for alternate life.

“Looking Down” is studying the Earth from a different vantage

point. Weather satellites are taken for granted, just like GPS location, and services like DirectTV. Some satellites are used for disaster events, such as spotting and tracking forest fires or oil spills.

Sometimes, new knowledge is gained quite by accident. For example, an ex-student of mine had a Cubesat on the outside of the International Space Station, looking down into thunderstorms. It turns out, thunderstorms act as huge natural particle accelerators, and they generate gamma ray bursts that are not observable from the ground. It's all in the point of view.

The Math part will be pervasive across all Cubesat topics from design to data analysis. It might be a good place to present the binary system. The Engineering part is designing, building, and testing the Cubesat, according to Good System Engineering Practices. The software part involves the ground support system ("Control Center") as well as the Cubesat's onboard computer. This also includes the topics of data formatting and precision, and data processing, storage, and transmission.

I don't think that jumping directly into building a Cubesat is the proper approach. Rather, we need to get the students involved in an area of interest to them, and then see how this could be addressed.

Projects

This section contains a series of Projects that can be conducted by students concerning Cubesats. An assessment of the difficulty and cost is included.

1. Cubesat Visual model – this involves downloading a single page .pdf of a Cubesat in actual size. You print it on heavy card stock, cut it out and assemble it. The image is available here: http://www.space.aau.dk/Cubesat/kits.html Cost: $0. This results in something tangible, and focuses attention.

2. Next, have the Students download NASA's free International Space Station ap for smart phones. It shows you the current position of the ISS, and has a high definition camera, showing what can be seen from the station. It gives you some idea of what the Earth looks like from that point of view. Discuss how this data could be used. Cost $0.

3. Download and install the COSMOS satellite control software product from Ball Aerospace. Available here: cosmosrb.com. Cost: $0. It can comfortably run on an old pc or laptop. It does require the Linux operating system, not Windows. I prefer the Ubuntu distribution. Some prior experience with Linux is good. The nice thing about COSMOS is, you don't need it to be talking to a spacecraft to use it. It has simulated spacecraft data built in, and gives the student a feel for being in front of a Satellite Control Center Console during a mission. Cost $0.

4. Once we have defined several student interest areas such as Weather, climate change, astronomy, we develop student teams to address these areas, based on mutual interest. Each team has a selected project, and designs its own Mission logo, which is printed in color, and put on the students' engineering notebooks. Cost: $3-5 per student for an engineering notebook. I like the Graph Composition style. Each team should also have a flash drive to hold software and documentation. Cost: $5.

5. Go over the System Engineering Process. (provided slideset) as an approach to developing the Cubesats. Cost: $0.

6. Build Cubesat prototypes. These will be non-flight models. Either purchase or 3-D print Cubesat chassis. These will basically be for fit-testing, and to remind every one what the focus is. A Bill-of-Materials will be supplied, A basic list is a RaspberryPi or Arduino computer, Lithium-ion

batteries, SD memory cards, and project-specific sensors.

7. Download and install the Satellite Toolkit (STK) software on a Windows pc or laptop. This is a free product from (https://www.agi.com/products/engineering-tools). It is an advanced modeling and simulation tool that includes streaming imagery and 3-D terrain models of Earth. It has visually attractive, moving screens. This will provide hours of interesting time, again, getting used to a different point of view. A large, wall-mounted monitor is good for this step.

Up to this point, we have spent little money. The next step is optional, but gets us more real, but requires some money and technical skills. One such product is the SAT-NOGS unit from satnogs.org. Sat-Nogs comes from the Technical University of Athens, Greece. They have open source construction plans and directions. The assembly can be built from readily available material such as pvc pipe and copper wire for around $500. This gets us into the area of Amateur Radio (Hams). You need some one with an Amateur Radio license to transmit, but not to receive. Initially, we will listen, eavesdrop really, only. The SatNogs concept is impressive. Each ground station agrees to receive any one's data, and put it up on a website. So, as the Network grows around the planet, better coverage is provided. You are not restricted to just getting a few minutes of communication with your Cubesat when it is overhead.

The plans and materials are for a ground station are online, and it is intended for students. The antenna is usually situated on the roof of a building, for better sky visibility. It uses a steered antenna. The details and documentation is available here: https:sdatnogs.org. Cost: $300-500.

The SatNogs electronics uses a RaspberryPi computer as a “receiver” using Software-Defined Radio, and an Arduino processor for steering the antenna via two small motors.

The next step is to build some prototype Cubesat hardware and test gear. This will include the Cubesat computer, sensors, and the communication link to the ground/test station we set up earlier.

An approach I have used successfully involves identifying students with similar interests and forming teams to implement their projects. An ideal team size is around 3-5. At the higher levels, these teams self-organize. It will probably be necessary for some teacher organization at the lower grade levels. It is essential to have each team member contributing, and learning from other team members. The teacher is an ex-officio member of all the teams/ The teams select a project lead, and assign roles and responsibilities.

The heart of the Cubesat is its onboard computer. This is typically a RaspberryPi board which is inexpensive, small and light, and very capable. It is built around the ARM 32-bit architecture. It can run the Linux operating system, various real-time operating systems, or you can use it in simple cases without an operating system. It can be programmed in the c language, or Python. It is actively used on Cubesat missions, in spite of its vulnerability to radiation damage (which is an advanced topic).
The Pi can be used to handle all the work, or it can be aided by a second small computer such as an Arduino. These are available in simple 8-bit or more capable 32-bit versions.

Once we have decided on a mission, we can select sensors for our Cubesat. These might be for visible light, infrared, or ultraviolet. They might include sensors for temperature, or magnetic field. Define the requirements, define the sensors needed. Might be a lesson there. Conversely, why aren't we measuring relative humidity in space? Trick question.

Now we construct the FlatSat – the Cubesat electronics spread out on a table for ease of access and testing. We include the battery, the computer board, and associated sensors. This will be connected

to the COSMOS machine via a wired connection, bluetooth, or wifi.

It is valuable at some time to have a 3D printer in the engineering lab. This provides a quick, relatively inexpensive way to turn students' concepts into tangible structure. This frame can then hold the computer, sensors, and battery. Open Source design software is available, and design files can be uploaded to a 3-D printing service for implementation. In the long run, investing in a 3-D printer will pay benefits later, as it is used in other projects. If the student can envision it, it can be printed. This is an awesome capability to have.

We can break the Cubesat down into two main parts – the spacecraft part, consisting of the power, control, radio, and structure, and the science payload. Essentially, the first part, referred to as the bus, can be common across missions, supporting different science instruments (payloads). This is great for Cubesats, as all the buses can be built the same. However they must support the science requirements for pointing, stability, electrical power, and data storage and transmission.

Breaking into Cubesats, pre-K to 12

Certainly, Cubesats can be introduced at the pre-K level, and NASA has age-appropriate material available for that. It is important, besides watching fantastic space videos, to get the students involved in projects, even if they just involve cardboard models. A globe is a good tool to show the relationship of the planet with the cubesat. A solar system model is a good way to introduce the other planets.

Topics

This section discusses the various topics that may be included in a STEM program, based on CubeSats.

Although STEM schools will have in-house expertise in Science (Physics, Chemistry, Biology), and Math (counting numbers through calculus), they are not heavily into Technology and Engineering. That's where you can ask for help – there's a lot of resources and knowledgeable individuals available out there. Reach out to local aerospace companies, or a local NASA Center for help. This book assumes a K-12 Program targeted to learning about cubesats, and building and exploring some hardware. It does not assume you have a flight project. That has been accomplished at the high school level. To go the next step to flight, see my books, "Cubesat Engineering" and "Cubesat Operations."

Within each topic area, we can pose questions like,

Science

- What is the metric system?
- What is Space weather?
- How do we measure and use the Earth's magnetic field?
- How can we use remote sensing to track storms, wildfires, oil spills, whale migration, etc?
- Is astronomy better above the atmosphere?

Technology

- How does a rechargeable battery work?
- How do we recharge the battery in space?
- What is the temperature of the Cubesat in orbit?
- How do we keep the temperature of the Cubesat within limits?
- How do we calibrate the sensors?
- What is involved in data formatting?

Engineering

- What is Electricity?
- What is Voltage, Current, Resistance, and Ohm's Law?

- What is a Circuit?
- What do we mean by Polarity?
- What is Digital Logic and logic levels?
- Serial versus parallel data transfer.
- Analog vs. Digital – compare and contrast.
- What is heat, and how do we measure and control it?
- How long will the battery last? How long does it take to recharge?
- How do we keep the Cubesat pointed in the right direction?'
- How do we test the Cubesat?
- If we drop the Cubesat off the table, will it still work? Let's find out! What broke?

Math

- What is the binary system, and how does binary math work?
- How do we convert between binary and decimal?
- What is Scientific and Engineering notation, and the range and accuracy of numbers?
- How do we convert sensor readings to engineering units?
- What are the error sources, and how can we get around them?

Systems Engineering Process

The system engineering process is a methodology of implementing complicated projects. It starts with the enumeration of requirements – what is it that the system has to do? This applies at the systems level, and will be allocated to parallel paths in the software and hardware. Design methodologies involve a plan and a process for creating a system to meet requirements. The more complex the system, the more complex are the plans. The process is the key. This applies equally to aqueducts, cathedrals, and space missions.

The design flow describes the sequence of steps in implementing the design methodology. There are various industry standard methodologies and models that have proved useful in large and complex projects. This is an outgrowth of the development of Systems Engineering in World War-II.

We apply a sound system engineering process to the project. We start by defining our requirements, which will mostly derive from the specifics of the environment that we want the Cubesat to operate in and observe. We will be cost and time constrained, which will drive the choices for build or buy. We can keep the cost low and the schedule intact if we use a lot of existing parts, and try to avoid custom builds, if possible. Keep it simple, and don't re-invent the wheel.

We can develop a set of “strawman” requirements for our project. To these would be added the requirements from the specific environment domain the Cubesat has to operate in. As the project proceeds along, the requirements “firm up.” (are better defined).

Requirements

Collecting the requirements, we are defining, “What is the question? What are we trying to do here?” Before we go very far down the path of implementation, we should have a good idea what exactly it is we are trying to accomplish. That should be written down, agreed to by all participants, and reviewed by the “Customer,” the teacher.

The requirements flow-down into more detailed levels, such as the functional requirements, the safety requirements, the interfacing requirements, the security requirements, and such things as size, weight, power, waterproof-ness, radiation tolerance….etc. And some of these requirements may be at odds with others. Delivery time is a requirement. So is maintainability. Sometimes, even color.

The more time we spend on the requirements, and their ramifications, the easier the rest of the task will be. What is the projected Mission duration? Do we anticipate unattended operation? Will our communications be upon opportunity or continuous? What is feasible for the radio link?

Consider Testability – How can we test and verify the Cubesat system? Can we test it in a simulated orbital environment. As a side project, should we build a thermal-vacuum testing chamber? How much redundancy do we need and can we support? Redundancy involves extra size, weight, and power.

Once we get our essential requirements together, we can define certain derived requirements from these. This would include factors such as:

Size, weight, power source, platform, mobility options, computing resources, communication resources, environmental: thermal range, moisture, radiation exposure, etc.

Specifications

The specifications are derived from the requirements. We need to address every requirement, but not include anything else. Every specification needs to be traceable back to a requirement, and result in a definition of what it is we're implementing, and a way to test that we have met the requirement.

When we do the specifications, we are answering the question,. "How will the requirements be met?" At this point we should have a complete and correct set of requirements, or there will be a lot of fixing up to do later, when it is a lot harder.

We need to determine if some of the requirements are mutually exclusive. If so, we have to go back and revisit the requirements. You can't have it both ways. We need to cross-reference specifications to requirements. There are automated tools to do

this for larger projects. A piece of paper works well for smaller projects.

Usage scenarios help with defining the requirements. Think about how you will use the Cubesat. This leads to operational planning. The mission needs to be monitored 24x7, but not necessarily by the users. We can use something like the COSMOS software to monitor parameters against pre-defined limits, and alert us when there is a problem. That assumes that we have defined these limits correctly.

Our goals in implementation are:

- Functionality – it has to work. For how long?
- Schedule: Time to deliver.(Semester-based)
- Build the unit.
- Minimize Non-recurring costs. Re-use facilities.
- Minimize manufacturing cost. Maximize re-use.
- A testable unit.
- Achieve correct Size, weight, power, energy, color, radiation tolerance, per spec. On time. On budget.

Reviews

Design Reviews are another tool to help us get the specifications correct, complete, and in sync with the requirements. You can use an independent review team to look over the project, and give recommendations. The students should do the project presentations. There might be a multi-teacher "panel of experts." At this point, technical details are less important than good presentation skills. NASA uses a series of reviews along the path to a successful project. There might be a Concept and Feasibility Review, a Preliminary Design Review (PDR), a Critical Design Review (CDR), a Flight Readiness Review (FRR). Each marks a milestone along the system engineering process.

Design Decomposition

There may be multiple ways to implement what we want to do, so there are design trade-offs to be made. What functions are done in hardware, and which are done in software? What is the interface specification – what are we talking to, and how are we talking? What are our data sources and users? How do we test and verify that the implemented system meets the requirements.

Will the system need to be changed or improved after completion? (duh!) Yes, Yes, it will. And, then, retested. And we want to fix more than we break.

Experience counts: To the person with a hammer, all problems resemble nails. Are you more software oriented, or more hardware oriented? Mechanical? Experience is often a major factor in implementation choices. There are multiple correct approaches.

Our obvious design goal is to construct an implementation with the desired functionality. A major design challenge is optimizing multiple design metrics simultaneously. Design metrics are a measurable feature of a system's implementation. Common metrics include:

- NRE cost (Non-Recurring Engineering cost): The one-time monetary cost of designing the system, including tools, both hardware and software.

- Size: the physical space required by the system – this is set by the Cubesat spec.

- Performance: the execution time or throughput of the system to get its job done.

- Power: the amount of electrical power consumed by the system (and the heat produced) in various modes of operation.

- Flexibility: the ability to change the functionality of the system easily.

- Time-to-prototype: the time needed to build a working version of the system. Ideally, this fits a semester schedule.

- Maintainability: the ability to modify the system during test and in orbit. What is the procedure for this?

- Correctness, and being able to prove it.

- Safety – there should be no hazardous materials, mechanisms, or scenarios.

- Security – only authorized Team members should have access to the equipment, parts, and facility.

- Ease of learning and use. Document early, document often. Develop good technical writing skills. Can the next semester's team pick up where the previous one left off?
- Testability – can you get enough data to figure out what happened when things go bad?

Design metric interactions are common - improving one may worsen others. Are these unintended consequences or blessings in disguise? There are trade-offs to be made. This is, after all, an engineering project.

System Architecture

The system architecture has two major components: the software and the hardware You are free to choose either the hardware base, the software environment, either, or neither. Keep in mind, software is hardware dependent. Software has the weight advantage, and is easier to change.

The hardware platform architecture, we'll assume for the moment, contains a cpu, memory, I/O devices, perhaps a communications bus to tie the I/O together. A single chip solution will give you cpu, memory, and I/O in one package. The software architecture will be constrained by the hardware, and will be limited by considerations of performance, testability, maintenance, and experience base.

Hardware and software architectures are interlinked by considerations based on the requirements; chiefly, speed, throughput, memory size, and I/O capacity. For example, it takes more cpu time to process and transmit image data that temperature samples.

The hardware platform can be prototyped on an inexpensive evaluation board. It will include the classical elements of cpu, memory, and I/O, plus clock circuits, a power supply, and usually a prototype area. In some cases, the prototype board can be used in the final design, if cost and schedule is an issue.

The processor choice doesn't revolve around operations per second, or word width so much anymore. We can get a 32- or 64-bit processor with a clock in the multi-gigahertz range cheap. As we tackle more elaborate problems, even these will seem inadequate. We need to focus more on integrated functionality, such as the number of I/O's, analogs, and interrupts. We also need to consider power consumption and heat generation as well. Can the cpu be put into a *sleep* mode to minimize power usage?

The implementation software language is a factor, but is overshadowed by yours and your team's experience. Choose a computer language you are good at, and they hit the ground running. C and Python are good choices. You can develop good software skills, or, at least, stay 1 lesson ahead of the students.

The development platform and its requirements is a minor issue, as we can assume it runs on a pc. It may be open source, or proprietary, and there may be multiple options available. You'll

need a dedicated development and test computer host.

Power for the embedded computer can be an issue. The systems will not be plugged into the wall. It will be solar powered. We have to factor in how long the Cubesat will be in sunlight versus how long it will be in darkness during a typical orbit. This makes for a very easy test. A light source on a timer, in a room with the system. Turn off the main room lights, and see how long the Cubesat operates, with its simulated Sun.

Keep in mind generated heat depends on power consumption, but battery life depends on energy consumption. Energy use is the power use, integrated in time. If we generate too much heat, we have to dump it somewhere. This might be a good thing. For a robotics project going to the Greenland Ice Shelf, we used the waste heat from the computer to keep the batteries warm. Power consumption is proportional to the voltage, squared. We might implement toggling; cycling the system on and off. More activity means more power

Redundancy

Redundancy refers to the technique of having multiple copies of critical components. This applies to hardware or software. This increases the reliability of the system but also the cost, complexity, and power usage. Redundant units can be deployed in parallel, such as extra structural members, where each unit can individually handle the load. This provides what is referred to as a margin of safety. An improvement in reliability can be achieved by simply adding a second unit in many cases. Of course, the second unit can also fail.

Safety

This area should address personal protective equipment such as goggles, lab coats, hearing protection, gloves, maybe hard hats. Give orientation and safety instructions on all powered tools. Where is the fire extinguisher? Luckily, we'll mostly be using low-

voltage, battery electronics.

Security

Are you familiar with the term, Cubesat-jacked? Well, I just made that up. But, it refers to a situation in which some one else takes over your project remotely.

All embedded systems have aspects of security. Embedded systems on Cubesats operate in an unfriendly world. They are vulnerable to variations of hacking, viruses and malware, theft, damage, spoofing, and other nasty techniques from the desktop/server world. GPS systems can be hacked to provide incorrect location or critical time information. Cell phones and tablets are connected wirelessly to large networks. A bored teenage hacker in Europe took over the city Tram system as his private full-scale railroad, using a TV remote. What about the teenager in an internet café is a third-world country. Can he take over and play with your Cubesat?

Some of these issues are addressed by existing protocols and standards for access and communications security. Security may also imply system stability and availability. Standard security measures such as reviews and audits, threat analyses, target and threat assessments, countermeasures deployment, and extensive testing apply to the embedded domain. Hang out with your institutational IT guy or gal for a while. They deal with this stuff every day.

A security assessment of a system involves threat analysis, target assessment, risk assessment, countermeasures assessment, and testing. This is above and beyond basic system functionality.

The completed functional system may need additional security features, such as intrusion detection and data encryption All of these additional features use time, space, and other resources that are usually scarce in small embedded systems.

Security has to be designed in from the very beginning; it can't just be added on. Memorize this. There will be a quiz. This was told to me by a guy that works at an agency that doesn't exist.

Mission Definition

First, Invent a cool project name, maybe an acronym. A key way to gain interest is to have a really cool name. Ask NASA. They use them to get Congressional budget approval all the time.

Define Look up/Look down. Essentially, for anything in Earth orbit, looking up defines an astronomy project, and looking down defines an Earth Science project. We can see things differently from space, and see different things. Many discoveries from space have involved things that are not visible, or not obvious, from the ground. I worked on a project that measured the salinity of the Ocean from orbit.

Weather and environmental observations are vastly improved our knowledge of the Earth's weather dynamics, and have improved our weather forecasting and storm tracking. From what we have learned from observing out own planet, we can apply some of this knowledge to the study of the environment of the other planets and their satellites. This ties into the astronomy subject area. Mars has a weather satellite, for notifying the surface rovers of dust storms. Venus, beneath the cloud cover, is in environment run-away. Greenhouse gas problems on a planetary scale. Some Venus lander spacecraft have melted. Yet it is almost the same size as the Earth, and just slightly closer to the Sun. What happened here, and how can we avoid it? There are no (longer) any client change skeptics on Venus.

Solar observation is a key topic. We can observe the very dynamic nature of our sun, which influences our weather. There are a series of monitoring satellites between the Earth and the Sun. Any radiation emitted from the sun (this includes visible light) reaches the Earth 8 minutes, 20 seconds later. Any particle emissions will reach Earth in 3-5 days. Massive solar storms create Coronal Mass Ejections (CME), which send a huge amount of very energetic

particle to Earth. The low energy ones create the Aurora. The more energetic ones kill satellites, and shut down power grids.

In the system Engineering process, applied to Cubesat and most any engineering project, there are several well defined steps.

Trades

Once your team has defined the mission, they can choose the appropriate sensors. A sensor might include a digital camera. Sensors that are available off-the-shelf include visible light, ultraviolet, infrared, magnetic, etc

Testing

First we need to understand the launch and on-orbit environment, and then we can define testing to show that our Cubesat can operate in that environment. One of Dr. Sandy Antunes' e-books on cubesats (listed in the bibliography) discusses a low cost approach to spacecraft testing.

We want to do several tests in the areas of thermal, mechanical, environmental, and functionality.

For cold cases, we can use a freezer. It will not get as cold as it will on orbit, but there's probably a 'fridge' handy. We can also use dry ice to get the temperature lower, but this introduces a safety concern.

For hot cases, we can use an infrared lamp. One test would be to see how much the side of the Cubesat facing away from the lamp goes above ambient. Again, caution. Maybe some yellow caution tape around areas where it gets dangerously hot.

Testing in the vacuum environment. A pressure cooker plus an automotive break vacuum pump can be used to build a cheap vacuum facility.

Vibration – An oscillating sander with a speed control makes a good vibration test device.

The mechanical testing is termed, “shake and bake.” The Cubesat is shaken, vibrated, and subject to acoustics – play loud music at it?

Cheap centrifuge – spin the payload around on a rope – outside.

Drop test – survival test. Onto a hard floor, down a stairwell, off the roof?

Monitoring the system during test can be a challenge. There is a special test data interface called JTAG which allows a computer to monitor a system under test. There is a special connector for the system under test, and it usually connects to a pc via USB. Software is then run on the pc to capture and display what the system under test is doing. The interface is inexpensive, and the software is usually free.

Open Source versus Proprietary

This is a topic we need to discuss before we get very far into software. It is not a technical topic, but concerns your right to use (and/or own, modify) software. It’s those software licenses you click to agree with, and never read. That’s what the intellectual property lawyers are betting on.

Software and software tools are available in proprietary and open source versions. Open source software is free and widely available, and may be incorporated into your system. It is available under license, which generally says that you can use it, but derivative products must be made available under the same license. This presents a problem if it is mixed with purchased, licensed commercial software, or a level of exclusivity is required. Major government agencies such as the Department of Defense and NASA have policies related to the use of Open Source software.

Adapting a commercial or open source operating system to a particular problem domain can be tricky. Usually, the commercial operating systems need to be used “as-is” and the source code is

not available. The software can usually be configured between well-defined limits, but there will be no visibility of the internal workings. For the open source situation, there will be a multitude of source code modules and libraries that can be configured and customized, but the process is complex. The user can also write new modules in this case.

Large corporations or government agencies sometimes have problems incorporating open source products into their projects. Open Source did not fit the model of how they have done business traditionally. They are issues and lingering doubts. Many Federal agencies have developed Open Source policies. NASA has created an open source license, the NASA Open Source Agreement (NOSA), to address these issues. It has released software under this license, but the Free Software Foundation had some issues with the terms of the license. The Open Source Initiative (www.opensource.org) maintains the definition of Open Source, and certifies licenses such as the NOSA.

The GNU General Public License (GPL) is the most widely used free software license. It guarantees end users the freedoms to use, study, share, copy, and modify the software. Software that ensures that these rights are retained is called free software. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU project in 1989. The GPL is a *copyleft* license, which means that derived works can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses are the standard examples. Copyleft is in counterpoint to traditional copyright. Proprietary software “poisons” free software, and cannot be included or integrated with it, without abandoned the GPL. The GPL covers the GNU/linux operating systems and most of the GNU/linux-based applications.

A Vendor’s software tools and operating system or application code is usually proprietary intellectual property. It is unusual to get the source code to examine, at least without binding legal

documents and additional funds. Along with this, you do get the vendor support. An alternative is open source code, which is in the public domain. There are a series of licenses covering open source code usage, including the Creative Commons License, the gnu public license, copyleft, and others. Open Source describes a collaborative environment for development and testing. Use of open source code carries with it an implied responsibility to “pay back” to the community. Open Source is not necessarily free.

The Open source philosophy is sometimes at odds with the rigidized procedures evolved to ensure software performance and reliability. Offsetting this is the increased visibility into the internals of the software packages, and control over the entire software package. Besides application code, operating systems such as GNU/linux and bsd can be open source. The programming language Python is open source. The popular web server Apache is also open source.

NASA didn't have a policy on use of Open Source software for quite a while. There were misunderstandings and suspicion about Open Source by government agencies and commercial firms. In the early 2000's, that began to change, as organizations began to realize the advantages of open source and collaborative development and testing environments. The current NASA open Source License, version 1.3, can be found here: http://opensource.org/licenses/NASA-1.3

Cubesat OnBoard Computers

What does the OBC do?

The onboard computer of the cubesat monitors and controls the cubesat, collects the science data, and makes sure it get transmitted to Earth. It can accept commands from the ground, and does energy and thermal management. It may do device pointing and orientation, depending on the needs of the sensors. It can also store

timed commands from the ground, for later execution. For Cubesat onboard computers, two popular choices are the Raspberry Pi and the Arduino.

Raspberry Pi

The Raspberry Pi is a small, inexpensive, single board computer based on the 32-bit ARM architecture. It is targeted to the academic market. Some versions support using SD cards for storage. The Raspberry Pi runs the GNU/linux and FreeBSD operating systems. It was first sold in February 2012. Due to the open source nature of the software, Raspberry Pi applications and drivers can be downloaded from various sites. It requires a single power supply, and dissipates less than 5 watts. It has USB ports, and an Ethernet controller. It does not have a real-time clock, but one can easily be added. It outputs video in HDMI resolution, and supports audio output. Standard I/O includes 8 general purpose I/O lines, UART, I2C bus, and SPI bus.

The video connection won't be needed onboard, but the newer models do have an image processing pipeline that interfaces with a simple camera.

Arduino

The project began in Italy in 2005 to produce a device for implementing student-built design projects less expensively. Now it represents the major computer architecture worldwide for small student projects.

The Arduino is a simple open-source single-board microcontroller. The hardware consists of a simple open hardware design for the Arduino board. The software support includes a standard compiler and a boot loader that runs on the board, along with numerous libraries of code.

An Arduino board consists of an 8-bit Atmel AVR microcontroller

or an Atmel 32-bit ARM. An important aspect of the Arduino is the standard way that connectors are arranged, allowing the CPU board to be connected to a variety of interchangeable add-on modules called *shields*. Shields allow for interfacing with sensors and actuators, as well as general I/O. Most boards include a 5-volt linear voltage regulator and a 16 MHz crystal oscillator for the clock. Some designs dispense with the on-board voltage regulator. An Arduino's microcontroller comes with a boot loader that simplifies uploading of programs to the on-chip flash memory.

Arduino hardware is programmed using a language similar to C++ with some simplifications and modifications, and an Integrated Design Environment (IDE) hosted on a laptop. The Arduino IDE is written in Java, so it can be run on just about any platform. A colleague of mine develops his software on his Android phone with an ap, but that's a little over the top. The Arduino IDE is designed to introduce programming to newcomers unfamiliar with traditional software development. It includes a code editor that is capable of compiling and uploading programs to the board with a single click. There is no need to edit makefiles or run complex programs on the command line. Generally, the Arduino is run without an operating system.

Cubesat Onboard and Support Software

To develop the software for the Cubesat, we will need a dedicated pc with the right software tools. Generally, the required tools are free and open source. In the general topic area of “Don't reinvent the wheel, again” check out what software is already written and available.

Now its true that sometimes it is just easier to code it yourself rather than try to understand what some one else did, but it does provide an option.

c and Python languages

The c programming language is general purpose, and maps well to

the architecture of many machines. It requires a *compiler*, a program that takes the c program you wrote (“source code”), and transforms it into ones and zeros that the computer will understand. It allows you to define data items in different formats, and do jumps and branches in your code, as well as data dependent branches (branch on zero).

It is an ANSI standard, and, like most languages it has nouns (variables) and verbs (action words). Several references books on the c language are included in the bibliography of this book. You can master the basics of programming in c in just a few hours on the computer. Getting good at it takes longer. You might also encounter a variant, called c++, which is more complex, but also more powerful. For the little embedded systems boards used in the Cubesat, a pc-class machine will be needed for developing software. Then, a usb connection can be made between the little computer and the pc, to download the code. This USB cable can then be used to see messages from the embedded board.

The Python language is particularly easy to use. It is an interpreted language, meaning there is an interpreter program running on the embedded computer, and it does not need to be compiled. It is open source software.

A couple of reference works on both c and Python are included in the Resources section. Keep in mind this is a Cubesat course, and if you try to teach programming, you'll run out of time. Find out who the good programmers are, and let them teach the others.

Sensors

Once we have defined a mission, we can look at the various sensors we will need to collect data. We define what to sense, and how to sense it. Later, we tackle error sources in sensors, limitations, etc. We need to see what interface the sensor needs, to hook it up to the embedded computer. It will be either analog (continuous voltage) or digital, (2 discrete voltages such as 0 and 3.3 v). It the sensor has an analog signal we need to hook it to one

of the analog to digital inputs on the computer. This will automatically convert the analog signal to a digital value the computer can work with. Check the allowed voltage range on the a/d inputs, so as not to exceed these. Some sensors are passive, and some are active, and need power. A temperature sensor might be as simple as a variable resistance in response to temperature changes. Sensors from the embedded board manufacturers will be designed to interface with their boards via an industry standard, such as i2c. The vendor will usually provide a pointer to code to read in the sensor values, and convert these to "engineering units" - a temperature, for example.

A *sensor* is a device that measures a physical quantity by changing state in response to the stimulation, and producing a signal. It is an analog world. It is rare that we get to interface directly to a digital source. Some sensors may indicate one of two states (presence/absence) with a simple digital signal that may only require voltage level shifting. Other signals, such as a switch closure, may appear digital, but require *denouncing* due to the physics of the actual contact, which actually closes and opens hundreds of times on activation. This is a form of signal conditioning for the sensor. We haven't yet considered voltage levels, current requirements, timing, and all those other real-world interfacing issues. We tend to view sensors as a "black-box" function, where the output is a valid representation of the applied signal. The ugly truth is, sensors are real-world devices that have their own non-linearity, parametric shifts, and they tend to respond to a lot more than the parameter we are interested in.

Passive sensors simply collect energy from the sensed phenomena; active sensors require power, or an excitation signal. A *transducer* is a device that converts one form of energy to another; a solar cell is an example. In the literature, the terms sensor and transducer are often used interchangeably.

All sensors are built to operate within a specified environment that corresponds to the temperature limits and other environmental conditions of its applied surroundings. Sensors may not satisfy all

essential conditions to operate within the system, including operating life, sensing range, accuracy, redundancy, low energy consumption, environments, mounting mechanism, reliability, sensing rate with response time, volume, and mass.

Signal conditioning refers to processing the sensed signal into a form from which the digital processor can then extract useful information. This may involve amplification or attenuation, analog to digital conversion, filtering, format conversion, electrical isolation, and other techniques. Noise filtering is a commonly applied technique. Sensors exhibit lag and hysteresis, which is a difference in offset from one measurement direction to another. Bias refers to the situation when the output is not zero when the measured quantity is. There can be dynamic errors, caused by rapid change in the input. *Drift* refers to the fact that, over time, the sensor may change output while the input remains steady.

The physics of the sensor must be considered. A relative humidity sensor measures relative humidity, but also temperature. A digital compass also reacts to magnetic fields produced by nearby wiring. Sensors are inherently non-linear. All of these characteristics must be understood and compensated for in software or hardware. With smart sensors, this compensation and processing would be accomplished within the sensor unit itself. For a simple sensor unit, some processing and conditioning must be done within the main embedded processor. Consider issues of operating life, range, maximum and minimum, accuracy, redundancy, energy consumption, heat generation, electromagnetic interference generation, electromagnetic interference susceptibility, mounting, reliability, sense rate, transient and steady-state response time, mass and volume, aging, and mean time to failure when choosing sensors.

The *MEMS*, or micro elector mechanical system, uses a chip-level integrated circuit technology to provide measurement devices. The advantage is that the sensors are made in processes developed for semiconductor manufacturer, and are inexpensive to mass-produce.

A typical accelerator sensor has a small gas-filled chamber with a

center heating element, and four temperature sensors around the edge. A static sensor results in all four temperature elements reading the same value. As the sensor is tilted, the higher sensor will read hotter. The sensor itself translates temperature differences into pulse widths. Some trigonometric calculations are needed to resolve angles. It also does not work in zero G.

Most cell phones have sensors can be used to determine acceleration, tilt, rotation, vibration, and other derived values.

Power Sources

Unless we have a really long extension cord, we will be relying on batteries and solar cells for power.

Power management onboard is a big issue. The computer can monitor the energy usage from the batteries. We keep track of the state-of-charge, which, at entry to orbital darkness should be at 100%. If it gets below, say, 10%, the onboard computer needs to take action to shut power loads down. Even put itself to sleep, as the last action. It can be awaken by command when back in sunlight, when the ground realizes it is not responding. We should have a monitoring program that measures power into and out of the battery, and defines a certain state-of-charge. Over time, this calculation will show us the degradation of the solar cells and the batteries.

Thermal Issues

There is no convection cooling in space. Onboard the Cubesat, we have conduction heating or cooling to the outer surface. Then, its radiation cooling or heating. One side of the Cubesat, facing the Sun, will be really hot. The other side, facing space, will be very cold. We might have as much as several hundred degrees difference across the unit. What temperatures can the electronics stand? How can we normalize the heat across the unit. (A classic maneuver that Apollo used on the way to the moon is called the Barbecue.) The cubesat would normally be covered in a multi-

layer insulating material. We can simulate some of that with plastic foam and aluminized Mylar. Add a heat lamp and a thermometer, and we can characterize the thermal properties of our Cubesat. Bring up a test rig that will record the temperature every few seconds, and log it. Later, dump the data to the MatLab software and do a graph.

Mechanical issues

In zero gravity, everything floats, whether you want it to or not. Floating conductive particles, bits of solder or bonding wire, can short out circuitry. This is mitigated by conformal coatings.

For mechanism such as movable solar arrays or pointing antennas, even the Cubesat deployment rails, there is a problem. Same or similar materials will weld themselves together in space. Most lubricants can't be used, as they evaporate and condense on optical surfaces.

The challenges of mechanisms in space are daunting, but much is now understood about the failure properties, and techniques to address them.

Telemetry and Command

We talk to and listen to the satellite payload via radio link. The satellite has limited power, and a small antenna, making the signal weak. This limits the rate that data can be sent. Also, in orbit, the payload is only overhead of our location for short periods of time every orbit. The general amount is 10 minutes per the low-Earth 90-minute orbit. How do we deal with that? What bandwidth do we need to get 90 minutes worth of data down in 10 minutes?

In the test rig, we can connect the Cubesat to the test computer via an Ethernet connection, or use wifi.

Amateur radio is involved in Cubesat work. They originally had their own satellites as communications relays in Earth orbit, way

before any one thought of Cubesats. These were all custom-built units. Now, if they can come up with the launch costs, each “Ham” operator can have his or her own relay satellite.

Data formatting – We have our choice on how we send commands to our Cubesat, and receive housekeeping and science data back. Here, again, standards are a good idea. We should choose a format that our ground/test system computer (COSMOS) recognizes. That makes every thing go smoothly. In addition, there is an advantage in keeping data in a format that is compatible with the computers we are using. The DOS file format for pc's running windows is a reasonable choice.

For flight use, there are standards such as CCSDS packets, which most missions choose. They are then compatible with a vast array of hardware and software. There is actually an Interplanetary Internet defined, which includes the planet in the identifier. Satellites in Earth orbit use a variation of a Mobile Data Protocol such as cell phone networks use. This is because rather than cabled to a network, or tethered by wifi, the system is moving (like a car down a highway), and switches from one antenna to the next automatically. Traditional protocols, developed for wired networks, don't work well in this mode.

File Systems

A file system provides a way to organize data in a standard format. The file system stores the data, and metadata (data about the data) such as date, time, permissions, etc. Some operating systems support multiple file systems. The Control Center software will take care of storing the incoming telemetry in a file system. It will also convert the bits to “engineering units” if you define the data format for it. For example, a temperature sensor will come with a graph of the relationship between its reading (in bits) and temperature (in Deg F or C). A calibration curve, its called.

The important thing to keep in mind about about a file systems is, don’t reinvent the wheel! There are many good file systems out

there, and the provide a compatibility across platforms. Most are based on the original disk operating system (dos) model. The Linux operating system has its own file system, but supports the Windows file system as well.

The legacy DOS file structure is built upon linked lists. The directory file contains lists of files and information about them. It uses a 32-byte entry per file, containing the file name, extension, attributes, date and time, and the starting location of the file on disk.

The File Allocation Table (FAT) is a map of allocated clusters on the disk. A cluster is the default unit of storage. It's size is a trade-off between efficiency of storage, and efficiency of access. A size of 256 bytes to 1024 bytes worked well in the early days. Two copies of the FAT are kept by the system, and these are on known locations of the storage media.

A directory file has entries for all of the files on the disk. The name of the file is in 8.3 format, meaning an 8 character file name, and a 3-character extension. The extension tells the type of the file, executable program, word processing, etc. By DOS convention, when a file is erased, the first character of the name is changed to the Hex character E5. The data is not lost at this point. If nothing else happens in the mean-time, the file can be un-erased, and recovered. However, the E5 signifies the space the file occupied is now available for use, and can be overwritten.

Learning experiences

Student Project/Competition – This is a favorite, because it is messy. Have the students design a protective shell for an egg, and then throw the projects off the roof. Which approach works, and which doesn't? And, Why?

Model Rocketry is a great tool to provide hands-on experience with payloads and the launch environment. It allows for testing

concepts in a real environment, and doing trades such as, “do I transmit the data, or store it onboard?” Just recently, a colleague who developed the Core Flight Software for NASA and does model rocketry at home, launched a very sophisticated payload. Unfortunately, it got stuck high up in a tree on the way down. He was able to download the data via wifi link, but the rocket is still in the tree. Sometimes, we just need a chainsaw to solve a problem.

Payloads for orbit are often tested first on high-altitude balloons and rockets, with programs run by colleges. Find out where, and talk yourself into a ride for your payload. NASA runs the *Rock-on* program, which puts student payloads into sounding rockets launched from the Wallops Flight Facility on the Virginia shore. The payloads are often recovered from the ocean. Several high altitude balloon flights are done as student projects. The University of Maryland has a program for student payloads.

Cubesats use the ride share option. They are mixed with other payloads and launched with a primary payload that covers most of the cost. The ugly secret is, the Cubesats replace sand ballast, that is used to get the center of mass of the rocket in the right place.

Dealing with Failures (Projects, not students)

Even NASA and the Big Aerospace Companies make mistakes. Aerospace failures tend to be large, expensive, and top news material. But, its important to learn as much as possible from failures; even to view failures as a learning opportunity. As we say, “It may be on fire, but it can serve as a bad example...” This is an important idea to keep in mind. We learn more from failures than from successes. It is necessary to build the mindset that every thing that goes not the way we intended is a learning experience. Documentation of failure cases is essential. Convening a group of participants and non-participants. to review the failure is a learning experience. We don't want to place blame here. The failure case is our best learning experience.

Ground Station

The ground station allows us to received data from our on-orbit satellite, and send it commands.

We need a radio receiver and transmitter. For the transmitter, we need at least some one with an Amateur radio license from the FCC. We are free to receive any satellite data we can. There are many options for the onboard radio system, from different vendors.

There is a large number of compatible ground stations around the world, operated by schools or individuals. These are all tied together, over the Internet. Each agrees to handle each other's telemetry and commanding. An example installation is at MakerFaire in New York. It features a yagi antenna for 144.8 MHz, and a helical antenna for 437 MHz. This is the SatNogs approach, discussed earlier

Case Studies of Cubesat Missions

They have been hundreds of Cubesat missions, of which, statistically, half worked. Here is a mention of a few missions. Cubesats can be found on most launches, these days.

Brazil's Tancredo-1

This interesting STEM project by middle school students was sent to the Space Station at the end of 2016, and deployed into space from the Japanese module. It was built from a kit from IOS, in California. It has the support of the Brazilian Space Agency, The National Institute for Space Research, and Unesco. It features a pre-recorded message from one of the school team members, that is continuously broadcast from orbit.

ZACUBE-1

The Cape Peninsula University of Technology in South Africa operates Africa's first Cubesat. It was built as a project of the South African National Space Agency, with assistance from the French

South African Institute of Technology. It was launched on a Russian rocket in 2013, and is collecting data on space weather. The project has resulted not only in on-orbit success, but 22 Master's degrees, 10 conference papers, and 3 journal papers. The project also spun off the African Space Innovation Center, with a Research Chair.

Indian multiCubesat

The Indian Space Research Organization (ISRO) launched a remarkable payload of 104 Cubesats from their Sriharikota launch site into Polar orbit. This was a new record for the number of satellites to orbit in one mission. It also validated the concept of Cubesats as a primary payload. They were deployed in 18 minutes. This changes the game somewhat. Usually, Cubesats hitch a ride with an expensive primary payload. Here, they were the primary payload.

Lunar Cubesats

Venturing out from Earth, out first destination is the moon. NASA's SIMPLEx mission (Small Innovative Missions for Planetary exploration) has 13 Cubesats lined up for a 2018 launch on an Orion vehicle. This will spend three weeks in space, including 6 days orbiting the moon. Cubesat missions will include the Lunar Flashlight, to look for water ice; the Near-Earth Asteroid Scout (with a solar sail); the BioSentinal, SkyFire, Lunar Ice Cube; CUSP, Cubesat for Solar Particles, and Lunar Polar Hydrogen Mapper. More will be selected closer to launch time.

All over the world, any nation can become spacefaring within a reasonable cost. Any school can potentially have its own space program. This ambitious project, to be launched in 2020, is an example. It will orbit the moon, and examine the effects of that environment on different life forms. The project is being done in cooperation with the UK Space Agency and ESA. Several other Cubesats will tag along for the ride. The Brazilian mission will

provide the communications back to Earth for all the associated missions. The launch will carried out by the European Space Agency (ESA). The UK Space Agency on the same flight will send the Pathfinder, the first commercial deep space mission.

It is a joint effort among top Brazilian universities: The National Institute of Space Studies (INPE), the Technological Institute of Aeronautics of USP, the National Laboratory of Synchrotron Light and the Institute of Technology of the Pontifical Catholic University of Rio Grande do Sul, with the University of Sao Paulo as the lead.

Marco Project

Mars Cube One (MarCO) is the first interplanetary cubesat mission, headed by JPL. It involves sending two 6-U Cubesats to Mars, along with the Insight Rover. The cubesats will separate at Earth orbit and proceed on their own. The mission was to be launched in March, 2016, when Mars was 1.07 AU distant, and arrive in September of 2016. The launch was postponed, however, due to a vacuum leak in the prime instrument. The mission is rescheduled for 2018.

The Cubesats will serve as a real-time communications relay with Earth during the critical descent and landing phase of the Rover. The Rover talks to the Cubesat relays over a UHF link, and the Cubesats relay this to Earth over an X-band link to the DSN. The Cubesat's X-band antenna is a large flat panel.

The Cubesats are stabilized with reaction wheels, and have propulsion systems to unload the wheels, and adjust their attitude.

PhoneSat

In 2013, NASA-Ames Research Center sent three special Cubesats up to the Space Station. These were special in that a off-the-shelf commercial grade cellphone was the onboard computer. The Phone-Sats reentered the atmosphere as planned, after

demonstrating the feasibility of the project. The cellphones' cameras were used for imaging, taking a total of one hundred pictures. . The phone was a Google Nexus One.

Mission Operations – Flying the Cubesat

Once the Cubesat Project is built, tested, and launched, we get into the details of Mission operations. From our Satellite Control Center, we are monitoring the telemetry, sending commands, planning the details of day-to-day operations, responding to anomalies, and such. It is a busy time.

However, we don't actually need a Cubesat in orbit to practice all this. We can use a Cubesat flatsat model, sitting on a bench, and communicating with the control center computer via ethernet, or even wifi. The control center software can easily fit on a laptop. We can use a large wall-mounted screen second monitor to give more people visibility.

Another option is to use another laptop, running a Cubesat software simulator instead of the actual Cubesat. This allows us to set up Mission Control, and run mission simulations for student teams.

How do we talk to a Cubesat?

If its a prototype or test article, we can use ethernet, or wifi. If it is in orbit, we need a RF ground station. Processing this data on the ground is easier if we format the data correctly. We should think in terms of a data structure, ideally, homogeneous, all data words are the same size. There should be a header, a few data words sent each time, including a spacecraft identifier (serial number), date and time stamp, that type of thing. Then we have a fixed-size block of data, let's say temperature readings, battery voltages, and a checksum over the entire data packet. As long as we have this definition, we can implement software to extract the relevant data, save it, and display it. This is exactly what the Cosmos product does.

Going a step further, if we organize the data in an accepted industry-standard protocol, there will be a lot of “off-the-shelf” software that we can use directly. The COSMOS product lets you define upper and lower red and yellow limits for each. It then checks each data point, and gives you on the screen a cautionary (yellow) or serious (red) alert message.

Goals for the Cubesat Operations Phase are to have the students:

- Understand the processing of data from the Cubesat, and the command data to the Cubesat.
- Understand the architecture of the Cubesat Control Center, and how it is evolving.
- Understand the functions that the Control Center implements.
- Understand the Control Center's role in normal and contingency operations.
- Understand the role of the Control Center in the Integration and Test Phase, and for training.
- Understand the roles of the members of the operations support team, and how they interact.
- Be able to serve as a member of a Cubesat Operations Team.

The Control Center facility can be set up early, and then used for Integration and Testing of the Cubesat itself. The Control Center Operations represents a Group operations, with different students assigned to various roles. Ideally, they choose these roles themselves, based on their preferences and skills.

Early in the Mission definition phase, various Operations Concepts (Ops Concepts) are defined, and these lead to requirements. The Ops Concepts drive the architecture of the Control Center. There must be sufficient flexibility for the Ops Concepts to evolve, as the mission progresses. This is part of the planning process for the mission, and can proceed without actual Cubesat hardware.

The Control Center provides work space, data systems for

telemetry, command capability, and coordination of activities between subsystems. The Control Center is responsible for spacecraft planning and scheduling. It is usually operating 24 x 7. The Control Center becomes a busy place during launch and spacecraft anomaly or failure situations. It is the workplace for the flight operations team. Different personnel may inhabit the back room at different times and mission mission phases. It is important to have a defined control center “space” where the operations team can congregate. This can be a dedicated room, or a designated space within a room.

Functional areas of the Control Center include Flight/Mission Control (Systems level) Guidance, Navigation, and Attitude Control, Thermal monitoring, Electrical Power, Onboard Computer and Data System, Communications and RF, and Payload Operations. Each of these generally maps to a seat in the Control Center, with a Team lead, typically a Systems Engineer, in charge. These roles can be assigned to students, based on their interests and abilities.

Here are some of the tasks done in the Control Center:

- Mission Planning and Scheduling.
- Commanding the spacecraft.
- Onboard engineering support (housekeeping).
- Data archiving.
- Performance monitoring, fault detection, and diagnostics.
- Calibration.
- Contingency planning – what if...?

On a simple mission, one person is assigned to each task. Maybe we rotate role's, so every one gets a chance to try each position. This can be for a simulated mission, using COSMOS' built-in simulator. Its good to get the experience of sitting at the console. Let's set up a dedicated side of the room, with a U-shaped arrangement, and students with computers/laptops in identified

positions.

The Planning and Scheduling function is off-line, ahead of the time it will be needed. It results in a detailed, optimized schedule of operations, for a 24-hour period or longer. Thik of this as homework.

Engineering support, or housekeeping activities, are accomplished according to schedule, or in response to anomaly conditions conditions.

Tracking

Once launched, the cubesat will be tracked by radar. NORAD, the North American Aerospace Command, based in Colorado, tracks all detectable orbital entities, from large satellites to space junk, zombie-sats, and the larger pieces of debris, as well as near-Earth asteroids. Each item is assigned a unique ID.

NORAD puts all this up on a website for your convenience, in a standard format called the “two-line element” (TLE). This contains the Keplerian orbital elements, the set of data describing the orbit of anything around the Earth, for a given point in time (epoch). It is a legacy format form the 1960's, that still works. It includes two data items of 80 ASCII charters each (an IBM punch card format).

Here is the format and contents of Line 1 (courtesy, Wikipedia).

Field	Columns	Content
1	1	Line number
2	3-7	Satellite number
3	8	Classification (U = unclassified)
4	10-11	Internat. Designator, last two digits of launch year
5	12-14	Launch number of teh year
6	15-17	Place of launch
7	19-20	Epoch, last two digits of year
8	21-32	Epoch, day of year, and fractional portion of day

9	34-43	First Time Derivative of the Mean Motion divided by two
10	45-52	Second Time Derivative of Mean Motion divided by six (decimal point assumed)
11	54-61	BSTAR drag term, (decimal point assumed)
12	63	number 0
13	65-68	element set number, incremented for new TLE
14	69	Checksum, modulo 19

Here is the format of line 2:

Field	Columns	Content
1	1	Line number
2	3-7	Satellite number
3	9-16	Inclination (degrees)
4	18-25	Right ascension of the ascending node (degrees)
5	27-33	Eccentricity (decimal point assumed)
6	35-42	Argument of perigee (degrees)
7	44-51	Mean Anomaly (degrees)
8	53-63	Mean Motion (revolutions per day)
9	64-68	Revolution number at epoch (revolutions)
10	69	Checksum (modulo 10)

Need your data? Get an account with spacetrack.org, and get it on line.

You can also use this service:
http://www.celestrak.com/NORAD/elements/

Need to watch out for space debris? Go here: http://satellitedebris.net/Database/

As an advanced topic, Orbit calculations can be supported in the control center, or done by a support facility. Simple modeling tools such as STK (Satellite Tool Kit) can be used. NASA-GSFC has an

Open source tool called ODTBX.

A nice engineering, modeling, and analysis software is called MatLab, which runs on a pc. It is from MathWorks.com. There is a 30-day free trial, and a student and campus edition. They offer online tutorials in its use. It is actively being used in STEM classes.

TRL

The Technology readiness level (TRL) is a measure of a device's maturity for use. There are different TRL definitions by different agencies (NASA, DoD, ESA, FAA, DOE, etc). TRL are based on a scale from 1 to 9 with 9 being the most mature technology. The use of TRLs enables consistent, uniform, discussions of technical maturity across different types of technology. We will discuss the NASA one here, which was the original definition from the 1980's.

Technology readiness levels in the National Aeronautics and Space Administration (NASA)

1. Basic principles observed and reported
This is the lowest "level" of technology maturation. At this level, scientific research begins to be translated into applied research and development.

2. Technology concept and/or application formulated
Once basic physical principles are observed, then at the next level of maturation, practical applications of those characteristics can be 'invented' or identified. At this level, the application is still speculative: there is not experimental proof or detailed analysis to support the conjecture.

3. Analytical and experimental critical function and/or characteristic proof of concept.

At this step in the maturation process, active research and

development (R&D) is initiated. This must include both analytical studies to set the technology into an appropriate context and laboratory-based studies to physically validate that the analytical predictions are correct. These studies and experiments should constitute "proof-of-concept" validation of the applications/concepts formulated at TRL 2.
4. Component and/or breadboard validation in laboratory environment.

Following successful "proof-of-concept" work, basic technological elements must be integrated to establish that the "pieces" will work together to achieve concept-enabling levels of performance for a component and/or breadboard. This validation must be devised to support the concept that was formulated earlier, and should also be consistent with the requirements of potential system applications. The validation is "low-fidelity" compared to the eventual system: it could be composed of ad hoc discrete components in a laboratory

TRL's can be applied to hardware or software, components, boxes, subsystems, or systems. Ultimately, we want the TRL level for the entire systems to be consistent with our flight requirements. Some components may have higher levels than needed.

5. Component and/or breadboard validation in relevant environment.

At this level, the fidelity of the component and/or breadboard being tested has to increase significantly. The basic technological elements must be integrated with reasonably realistic supporting elements so that the total applications (component-level, sub-system level, or system-level) can be tested in a 'simulated' or somewhat realistic environment.

6. System/subsystem model or prototype demonstration in a relevant environment (ground or space).

A major step in the level of fidelity of the technology

demonstration follows the completion of TRL 5. At TRL 6, a representative model or prototype system or system - which would go well beyond ad hoc, 'patch-cord' or discrete component level breadboarding - would be tested in a relevant environment. At this level, if the only 'relevant environment' is the environment of space, then the model/prototype must be demonstrated in space.

7. System prototype demonstration in a space environment.

TRL 7 is a significant step beyond TRL 6, requiring an actual system prototype demonstration in a space environment. The prototype should be near or at the scale of the planned operational system and the demonstration must take place in space.

The TRL assessment allows us to consider the readiness and risk of our technology elements, and of the system.

And in Conclusion...

The National STEM program is key to the United States' and the World's dominance and application of advanced technologies. I can't say for a fact that every nation has Cubesat programs, but a lot do. Most never get to the launch phase. Along the way, students have learned a lot about engineering in general. Most importantly, we have gotten them interested and engaged. Even if they never do another engineering project in their life, they will understand the process, its applications, and its limitations. Some of them will go on to explore Mars, or find solutions to problems we are not yet aware of. Maybe these will be your students.

Cubesats are small simplified versions of satellites, that can be used in the Classroom to generate student interest, and as a springboard into complex topics. “Wait,” you say. I'm not a Rocket Scientist, I'm a teacher. Well, I'm not a Rocket Scientist either, I'm a Rocket Engineer. So, I can help you find and understand what you need to pull together a class, project, or curriculum. You can always ask the experts for help. We'll be here – unless we're going to Mars.

The Cubesat Design Specification

The Cubesat Design Specification, developed by California Polytechnic State University, defines the physical and interface specifications for Cubesats, and gives testing requirements for vibration, thermal-vacuum tests, and shock, as well as safety. Since a Cubesat flys with other Cubesats in a deployment device, and with a primary payload, safety is a concern. Cubesats are expected to have an on-orbit lifetime of less than 30 years.

Here is a synopsis of CubeSat requirements:

Mass

- Each satellite may not exceed 1 kg of mass
- The CubeSat center of mass must be within 2 cm of the geometric center.

Structure

All edges contacting rails must be rounded. Cubesats must have at least 75% (85.125 mm of a possible 113.5 mm) of flat rail contact with the deployer- To prevent cold-welding, raw metal is not allowed as the contact surface of the bottom standoff. Derlin inserts, or a hard anodize are examples of acceptable contact surfaces.
- The outer surfaces of the CubeSats are required to be hard anodized in order to prevent wear between the sliding rails and the

CubeSats.
- Separation springs (SSMD-51P recommended) must be included at designated contact points. A custom separation system may be used upon approval by CalPoly/Stanford launch personnel.
- One deployment switch is required (two are recommended) for each CubeSat.

Material

- The use of Aluminum 7075 or 6061-T6 is suggested for the main structure. If other materials are used, the thermal expansion coefficient must be similar to that of Aluminum 7075-T3 (the POD material) and approved by CalPoly/Stanford personnel.

Deployables

- A time delay, on the order of several minutes, must be present between release from the P-POD and any satellite hardware deployment, to allow for satellite separation.
- P-POD rails and walls cannot be used to constrain deployable hardware

Communication

- There must be a time delay, on the order of several minutes to an hour, before all primary transmitters are activated. Low power beacon transmitters may be activated after deployment.
- Operators must provide proof of the appropriate license for frequency use.

Power

- CubeSats with rechargeable batteries must have the capability to receive a transmitter shutdown command, compliant with FCC regulations.
- Satellites that require testing and battery charging must provide an external hardware interface to access the power/data port
- A 'remove before flight pin' is required to deactivate the CubeSats during integration outside the P-POD. The pin will be removed once the CubeSats are placed inside the P-POD.

General

- Absolutely no pyrotechnics are allowed inside the CubeSat
- Must be able to accommodate satellites with solar panels

mounted on the external walls- A final check of specifications will be conducted prior to launch.

There are emerging standards for larger Cubesats, such as 6U (12 kg, 12 x 24 x 36 cm), 12U (24 kg, 23 x 24 x 36 cm) and 27U (54 kg, 34 x 35 x 36 cm). These allow the canister to constrain Cubesat deployables such as antennae and solar array. In the original Cubesat specification, this task had to be handled by the Cubesat itself. Even as they get bigger, the standard architecture and modularity of the Cubesat remains a game-changing advantage.

The following Appendix covers some advanced topics, and are presented for background information.

Appendix- Orbital Radiation Environment and Effects

There are two radiation problem areas: cumulative dose, and single event. Operating above the Van Allen belts of particles trapped in Earth's magnetic flux lines, spacecraft are exposed to the full fury of the Universe. Earth's magnetic poles do not align with the rotational poles, so the Van Allen belts dip to around 200 kilometers in the South Atlantic, leaving a region called the South Atlantic Anomaly. The magnetic field lines are good at deflecting charged particles, but mostly useless against electromagnetic radiation and uncharged particles such as neutrons. One trip across the Van Allen belts can ruin a spacecraft's electronics. Some spacecraft turn off sensitive electronics for several minutes every ninety minutes – every pass through the low dipping belts in the South Atlantic.

The Earth and other planets are constantly immersed in the solar wind, a flow of hot plasma emitted by the Sun in all directions, a result of the two-million-degree heat of the Sun's outermost layer, the Corona. The solar wind usually reaches Earth with a velocity around 400 km/s, with a density around 5 ions/cm^3. During magnetic storms on the Sun, flows can be several times faster, and stronger. The Sun tends to have an eleven year cycle of maxima. A solar flare is a large explosion in the Sun's atmosphere that can release as much as 6×10^{25} joules in one event, equal to about one sixth of the Sun's total energy output every second. Solar flares are frequently coincident with sun spots. Solar flares, being releases of large amounts of energy, can trigger Coronal Mass Ejections, and accelerate lighter particles to near the speed of light toward the planets.

The size of the Van Allen Belts shrink and expand in response to the Solar Wind. The wind is made up of particles, electrons up to

10 Million electron volts (MeV), and protons up to 100 Mev – all ionizing doses. One charged particle can knock thousands of electrons loose from the semiconductor lattice, causing noise, spikes, and current surges. Since memory elements are capacitors, they can be damaged or discharged, essentially changing state.

Vacuum tube based technology is essentially immune from radiation effects. The Russians designed (but did not complete) a Venus Rover mission using vacuum tube electronics. The Pioneer Venus spacecraft was launched into Venus orbit in 1978, and returned data until 1992. It did not use a computer, but an attitude controller built from discrete components.

Not that just current electronics are vulnerable. The Great Auroral Exhibition of 1859 interacted with the then-extant telegraph lines acting as antennae, such that batteries were not needed for the telegraph apparatus to operate for hours at a time. Some telegraph systems were set on fire, and operators shocked. The whole show is referred to as the Carrington Event, after amateur British Astronomer Richard Carrington.

Around other planets, the closer we get to the Sun, the bigger the impact of solar generated particles, and the less predictable they are. Auroras have been observed on Venus, in spite of the planet not having an observed magnetic field. The impact of the solar particles becomes less of a problem with the outer planets. Auroras have been observed on Mars, and the magnetic field of Jupiter, Saturn, and some of the moons cause their "Van Allen belts" to trap large numbers of energetic particles, which cause more problems for spacecraft in transit. Both Jupiter and Saturn have magnetic field greater than Earth's. Not all planets have a magnetic field, so not all have charged particle belts.

Bibliography

Antunes, Dr. Sandy, *Surviving Orbit the DIY Way: Testing the Limits Your Satellite Can and Must Match,* 2012 , Maker Media,

Inc., ISBN-449310621.

Antunes, Dr. Sandy, *DIY Instruments for Amateur Space: Inventing Utility for Your Spacecraft Once It Achieves Orbit,* Maker Media, Inc., ISBN-978-1449310646.

Antunes, Dr. Sandy, *DIY Comms and Control for Amateur Space: Talking and Listening to Your Satellite,* 2015, Maker Media, Inc., ISBN-978-1449310660.

Antunes, Dr. Sandy, *DIY Satellite Platforms: Building a Space-Ready General Base Picosatellite for Any Mission,* 2012, Maker Media, Inc. ISBN-978-1449310608.

Asnuda, Paul A. "Open Courseware and STEM Initiatives in Career and Technical Education", avail, ir.library.illinoisstate.edu/cgi/viewcontent.cgi?article=1042&context=jste.

Cameron, Schyrlet, Craig, Carolyn *STEM Labs for Earth & Space Science, Grades 6 – 8*, Mark Twain Media , 2017, ISBN-1622236394.

Chan, Irene *Baby Loves Aerospace Engineering! (Baby Loves Science),* Charlesbridge, 2016, ISBN-1580895417.

Committee on Achieving Science Goals with CubeSats; Space Studies Board; Division on Engineering and Physical Sciences; National Academies of Sciences, Engineering, and Medicine, *Achieving Science with CubeSats: Thinking Inside the Box,* National Academies Press, 2016, ISBN 978-0-309-44263-3. avail. http://www.nap.edu/23503

Dugan, Christine *Space Exploration*, Teacher Created Materials, 2012, ASIN-B01J8W5WZ8.

Harrington, Eileen G. *Science and Technology Resources on the*

Internet, STEM Education in the United States: Selected Web Resources, in Issues in Science and Technology Leadership, Summer 2015, avail: http://istl.org/15-summer/internet.html

Kerber, Jonathas G. "An Introduction to Cubesats as Teaching Tools and Technology Testing Platforms, University of Ottawa

McCue, Dr. Camille *Coding for Kids for Dummies*, 2014, For Dummies Books, ASIN-B00YDK7ODO.

Melvin, Leland *Chasing Space Young Readers' Edition*, Amistad, 2017, ASIN-B01L6RD57K.

Moore, John D. "Integrating Small Satellites into the National K-12 STEM Education Discussion ," (2013): JoSS, Vol. 2, No. 2, pp. 201-211.
avail, http://www.jossonline.com/wp-content/uploads/2014/12/0202-Integrating-Small-Satellites-into-the-United-States.pdf.

Neuhauser, Alan "Space Cadets: STEM Program Gives Students Control of Satellites," U. S. News and World Report, Aug. 27, 2014.

Sosa, Kain A. "The New STEM Education: Why CubeSat Technology is the perfect vehicle to get out students ready for space exploration," Powerpoint presentation, avail: http://mstl.atl.calpoly.edu/~workshop/archive/2017/Spring/Day%203/Session%202/2_KainSosa.pdf

Stakem, Patrick H. *Cubesat Engineeering*, PRRB Publishing, 2017, ISBN-9781520754017.

Stakem, Patrick H. *Cubesat Operations*, PRRB Publishing, 2017, ISBN-9781520767178.

Stakem, Patrick H. *Interplanetary Cubesats*, PRRB Publishing, 2017, ISBN-9781520896779.

Stakem, Patrick H. *Cubesat Constellations, Clusters, and Swarms, Stakem,* PRRB Publishing, 2017, ISBN-978-1520767543.

Sweeney, Linda Booth, Meadows, Dennis The Systems Thinking Playbook: Exercises to Stretch and Build Learning and Systems Thinking Capabilities, 2010, Chelsea Green Publishing, ISBN-1603582584.

Resources

Programming – all available on Amazon.com

Mueller,John Paul *Beginning Programming with Python For Dummies*,1st Edition ISBN-1118891457.

Thompson, Joe *PYTHON: PYTHON'S COMPANION, A STEP BY STEP GUIDE FOR BEGINNERS TO START CODING TODAY!* 2016 ASIN-B01MDM1PE4.

Python: The Complete Python Quickstart Guide (For Beginner's) (Python, Python Programming, Python for Dummies, Python for Beginners), 2015, ASIN-B0118HRIBC.

Gookin,Dan *C For Dummies*, 2nd Edition, 2004, ISBN-100764570684.

Gookin,Dan, *Beginning Programming with C For Dummies,* 1st Edition, ISBN-1118737636.

Perry, Greg, Miller, Dean *C Programming Absolute Beginner's Guide* (3rd Edition,) Que Publishing; 3 edition,, 2013, ISBN-0789751984.

Here is a 3-D model of a Cubesat that you can download, print on heavy cardstock, and assemble.

http://www.space.aau.dk/Cubesat/kits.html

Here is a 3-D CAD model of a Cubesat that can be downloaded free from NASA, and printed. https://nasa3d.arc.nasa.gov/detail/cubesat

https://www.nasa.gov/kidsclub/index.html

https://www.nasa.gov/connect/apps.html

www.IrvineCubeSat.org – high school program in the United States to launch an operational Cubesat Mission.

https://www.dcstemnetwork.org/resource/cubesat-competition CubeSat Competition, DC STEM Network.

https://www.usnews.com/news/stem-solutions/articles/2014/08/27/space-cadets-stem-program-gives-students-control-of-satellites

Project Da Vinci, "Idaho Students Inspired by XPRISE to Make Spaceship," SPACEREF.com, Nov. 16, 2016, www.spaceref.com

"United Launch Alliance has a competition to launch college cubesats, with Outreach to stem." http://www.ulalaunch.com/cubesats.aspx

https://www.ardusat.com/ applicable experiments and classroom material.

NASA systems Engineering Handbook, NASA SP-2007-6105. Avail: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008301.pdf

Cubesat Design Specification, Cubesat Program, California Polytechnic State University, available:

https://www.google.com/search?q=Cubesat+Design+Specification&ie=utf-8&oe=utf-8

https://www.nasa.gov/mission_pages/

Planetary geology: a teacher's guide with activities in physical and Earth sciences, SuDoc NAS 1.19/4:1998-03-109-HQ), 1998, ASIN-B00010XCMC.

Teachers and Students Investigating Plants in Space: A Teacher's Guide with Activities for Life Sciences (EG-1997-02-113-HQ) 1997 ASIN-B000H48XM6.

NASA, *Rockets : a teacher's guide with activities in science, mathematics, and technology, K-12,* 1996, EG-1996-09-108-HQ, ASIN-B000111AX4.

NASA's Educator Resource Center, free, professional, equipment loan, publications. https://www.nasa.gov/offices/education/programs/national/ercn/home/index.html

NASA State of the Art Report of Small Spacecraft Technology, available online at: https://sst-soa.arc.nasa.gov/

Small Satellites for Secondary Students, www.s4.sonoma.edu; http:// s4.sonoma.edu/guide.html.

Space Station Simulator Educator's Guide, (grades 5-8), avail: https://www.nasa.gov/directorates/heo/education/ISS_workbook.html

https://www.nasa.gov/feature/wallops/2016/nasa-seeks-students-experiments-for-2017-edge-of-space-balloon-flight.

NASA Educator Resources, *Algebra 2 With Space Science Applications*, avail:

https://www.nasa.gov/audience/foreducators/topnav/materials/listbytype/Algebra_2.html

U. S. Department of Education, Resources for STEM Education, 2017, avail: https://innovation.ed.gov/files/2017/04/Resources_for_STEM_Education.pdf

Infant and Toddlet STEM; https://depts.washington.edu/cqel/PDFs/Presentations/Infant%20&%20Toddler%20STEM%20NW%20Institute%202014%20Powerpoint.pdf

Wikipedia, various.

A list of open source tools for Cubesats can be found here:

- http://wiki.developspace.net/Open_Source_Engineering_Tools
- https://www.nasa.gov/audience/foreducators/index.html (Sections for K-4, grades 5-8, 9-12)
- https://blogs.nasa.gov/ivv/tag/lego/
- http://earthobservatory.nasa.gov/Newsroom/BlueMarble.
- Free download of Satellite Tool Kit - http://www.agi.com/products/by-product-type/applications/stk/
- https://www.researchgate.net/publication/283359843_science_and_technology_resources_on_the-internet_STEM_education_in_the_united_states_selected_web_resources.

- STEM lessons from Space, avail: https://www.nasa.gov/audience/foreducators/stem-on-station/lessons

K-12 Cubesat

A new Stanford University Space Systems Design Laboratory CubeSat effort designed with kindergarten through 12th grade (K-12) education in mind.

The project is named KatySat, short for "Kids Aren't Too Young for Satellites," explained Ben Yuan, Project Manager. Next year is being eyed for a launch given success in fund raising. The price tag of the actual satellite is in the $10,000 range, but then there's the expense of launch--in the range of $40,000 per cube--and other related costs. See: https://www.totalphase.com/solutions/wp/stanford-ssdl/

---

http://www.space.com/1570-big-news-small-satellites-cubesats-rule.html

http://www.totalphase.com/solutions/wp/stanford-ssdl

http://teachers.egfi-k12.org/tag/cubesat/

http://www.cubesatkit.com/

http://kcs.kana.k12.wv.us/schs/CubeSat/Sopnsor%20Letter.htm

http://www.ulalaunch.com/cubesats.aspx

http://www.wvspacegrant.org/wp-content/uploads/2014/11/CubeSat-Direction-Booklet1.pdf

http://www.pocketqubeshop.com/hardware/pocketqube-kit

http://www.wvspacegrant.org/wp-content/uploads/2014/11/LEGO-STF-1-Challenge-Description-.pdf

http://www.stf1.com/stf1.php

http://www.dreamup.org/k-12

https://www.vssec.vic.edu.au/programs/launch-box/

https://www.ardusat.com

https://www.nasa.gov/centers/ames/engineering/projects/phonesat.html

https://www.nasa.gov/smallsat-institute

http://wiki.developspace.net/Open_Source_Engineering_Tools

https://www.edu.gov/stem

Commercial vendor: http://www.interorbital.com/

NASA West Virginia Space Grant Consortium, www.wvspacegrant.org/

https://www.nasa.gov/directorates/spacetech/small_spacecraft/phonesat.html

Infant and Toddlet STEM; https://depts.washington.edu/cqel/PDFs/Presentations/Infant%20&%20Toddler%20STEM%20NW%20Institute%202014%20Powerpoint.pdf

.

Glossary of Terms

1U – one unit for a Cubesat, 10 x 10 x 10 cm.
3U – three units for a Cubesat
6u – 6 units in size, where 1u is defined by dimensions and weight.
Actuator – device which converts a control signal to a mechanical action.
AIAA – American Institute of Aeronautics and Astronautics.
AIST – NASA GSFC Advanced Information System Technology.
AmSat – Amateur Satellite. Favored by Ham Radio operators as communication relays.
Android – an operating system based on Gnu-Linux, popular for smart phones and tablet computers.
ANSI – American National Standards Organization.
Antares – Space launch vehicle, compatible with Cubesats, by Orbital/ATK (U.S.)
AP – application programs.
API – application program interface; specification for software modules to communicate.
Apogee – the point of greatest distance from the Earth by an orbiting body.
APL – Applied Physics Laboratory of the Johns Hopkins University.
Arduino – an 8- or 32 bit architecture, generally ARM. CPU with memory and I/O.
ARM – Acorn RISC machine; a 32-bit architecture with wide application in embedded systems.
ASIN – Amazon Standard Inventory Number.
ARRL – American Radio Relay League – amateur radio.
ATS – absolute time sequence.
AU – astronomical unit. Roughly 93 million miles, the mean distance between Earth and Sun.
AVR – a family of 8-bit processors from Atmel.
Binary – 2 values, 0 and 1.
BIST – built-in self test.
Bit – binary variable, value of 1 or 0.

bootloader – initial code that gets a computer started, pulling it up "by its bootstraps"
BSD – Berkeley Software Distribution (of Unix). Software operating system.
Bluetooth – short range wireless networking.
Bootloader – initial program run after power-on or reset. Gets the computer up & going.
C – a programming language, widely used.
C++ - an object-oriented programming language
CalPoly – California Polytechnic State University, San Luis Obispo, CA., originator of the Cubesat.
CCSDS – Consultive Committee on Space Data Systems. (Standards organization)
CDR – critical design review
C&DH – Command and Data Handling
CDFP CCSDS File Delivery Protocol
cFE – Core Flight Executive – NASA GSFC open source flight software.
CFS – Core Flight System – NASA GSFC open source flight software.
Chip – integrated circuit component.
Clock – periodic timing signal to control and synchronize operations.
CME – Coronal Mass Ejection. Solar storm.
CogE – cognizant engineer for a particular discipline; go-to guy; specialist.
Computer – Does math and logical operations on data. Includes storage and I/O.
Cots – commercial, off the shelf
CPU – central processing unit.
CRC – cyclic redundancy code – error detection and correction mechanism.
Deadlock – a situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.
Derlin – a plastic material from DuPont.
DOF – degrees of freedom.
Downlink – from space to earth.

DSN – Deep Space Net – NASA set of three large antennae spaced around the globe, used for missions far away from Earth.
Ecliptic – the apparent motion of the Sun's motion on the celestial sphere, as seen from the Earth.
EDAC – error detection and correction.
EGSE – electrical ground support equipment
Embedded computer – small computer focused on sensing and control. Usually does not host its own tools.
Ephemeris – a data set of orbital position.
ESA – European Space Organization.
ESRO – European Space Research Organization
ESTO – NASA/GSFC – Earth Science Technology Office.
Ethernet – networking standard. Can be hard-wired, or wireless.
ev – electron volt, unit of energy.
FCC – Federal Communications Commission
FlatSat – engineering model, laid out flat on the workbench for ease of access.
FPU – floating point unit – performs the 4 basic math operations on data with a mantissa and an exponent.
Full duplex – communication in both directions simultaneously.
GEO – geosynchronous orbit.
GHz – giga (10^9) Hertz.
GNC – guidance, navigation, and control.
GND - ground
Gnu – recursive acronym, gnu is not unix.
GPL = Gnu Public License. Open source.
GPS – Global Positioning system – Navigation satellites.
GPU – graphics processing unit, like a cpu, but optimized for image and video data.
GSFC – Goddard Space Flight Center, Greenbelt, MD.
Gyro – (gyroscope) a sensor to measure rotation.
Handshake – co-ordination mechanism in hardware or software/
HDMI – High-Definition Multimedia Interface.
Housekeeping data – temperatures, battery levels, pointing, etc.
I^2C – a serial bus architecture to carry data.
IDE – Integrated Design Environment, development software to produce code for an embedded cpu.

I/O – input & output of data.
ISBN – International standard book number.
ISO – International Standards Organization.
ISS – International Space Station.
IT – information technology
IV&V – Independent validation and verification.
JOSS – Journal of Small Satellites
JPL – Jet Propulsion Lab, Pasadena, California. Operated by Cal Tech for NASA.
JTAG – Joint Test Action Group, debugging interface.
KatySat – Kids Aren't Too Young for Satellites – Stanford University, Space and Systems Development Laboratory (SSDL)
Lagrange Point – a null point in the gravity field in the solution to the restricted 3-body program.
LaRC – (NASA) Langley Research Center.
Lbf – pounds-force.
Led – light emitting diode. Various colors available.
LEO – low Earth orbit.
Let - Linear Energy Transfer
Linux – open source operating system, Unix-like.
LSP – (NASA) launch services program, or launch services provider.
Makefile – a file with a set of directives to automate a process.
Memory leak – when a program uses memory resources but does not return them, leading to a lack of available memory.
Memory scrubbing – detecting and correcting bit errors.
MEMS – micro electro mechanical system
MHz – megahertz, 10^6 cycles per second.
Microcontroller – a single board cpu, memory, and I/O, intended for sensing and controlling.
Microsat – satellite with a mass between 10 and 100 kg.
MLI – multi-layer insulation.
NanoRacks – a company providing a facility onboard the ISS to support Cubesats
NASA – National Aeronautics and Space Administration.
NEA – near Earth asteroid.

NEC – near Earth comet.
NEN – (NASA's) Near Earth Network.
NEO – near Earth object.
NOAA – (U.S.) National Oceanographic and Atmospheric Administration.
NORAD – North American Aerospace Defense Command (USAF) – tracks everything in space.
NRE – non-recurring engineering; one-time costs for a project.
NSF – (U.S.) National Science Foundation.
NSR – non-space rated.
NWS – (U.S.) National Weather Service
Open source – methodology for hardware or software development with free distribution and access.
Operating system – software that controls the allocation of resources in a computer.
Parallel – multiple bits at a time. Usually a power of two.
PHO – potentially hazardous object
PCB – printed circuit board.
PDR – preliminary design review.
Perigee – point of closest distance to an orbiting body of the Earth.
Phonesat – small satellite using a cell phone for onboard control and computation.
PI – Principal Investigator
Picosat – small satellite with a mass between 0.1 and 1 kg.
PiSat – a Cubesat architecture developed at NASA-GSFC, based on the Raspberry Pi architecture.
Poc – point of contact
PPF – payload processing facility
PPL – preferred parts list (NASA).
PPOD - Poly Picosatellite Orbital Deployer
PWM – pulse width modulation. Popular motor control technique, using variable length pulses.
PVC – pol vinyl chloride – a plastic material for structural shapes.
Python – a programming language,
RAM – random access memory; any item accessible in 1 clock cycle.
Raspberry Pi – a 32 bit ARM architecture processor, on a board

with memory and I/O.
RTC – real time clock.
RTOS – real time operating system.
RTS – relative time sequence.
RX – receive.
S4 – Small Satellites for Secondary Students.
SAA – South Atlantic anomaly. High trapped radiation zone.
Safe-hold – a satellite survival mode, involving shutting down all non-essential loads.
Sandbox – an isolated and controlled environment to run untested or potentially malicious code.
SATNOGS – Satellite Networked Open Ground Station.
SD – Secure Digital flash memory card.
SDR – software defined radio
SDVF – Software Development and Validation Facility.
Serial – bit by bit.
SEU – single event upset (radiation induced error).
Shield – in the Arduino world, a snap-on I/O board.
SN – (NASA's) Space Network
SOA – safe operating area; also, state of the art.
SOC – System-on-a-chip, the cpu, memory, and I/O on one chip.
Solar Sail – using the outflow of the solar wind for propulsion.
Spacewire – high speed (160 Mbps) serial link.
SPI – serial peripheral interface.
STEM – Science, Technology, Engineering, Mathematics
STEMSat - Ugandan Satellite TV provider
STEM-SAT – Cubesat for STEM.
STK – Satellite Tool Kit
STL – stereolithography Compact Shared Document (CSD) file, for 3D printing.
STOL – system test oriented language. A scripting language for computers to automate tests.
SWAP – size, weight, and power.
TARC – Team America Rocketry Challenge.
TDRSS – Tracking and Data Relay satellite system.
T&I – test and integration.
Train – a series of satellites in the same or similar orbits, providing

sequential observations.
Transceiver – transmitter and receiver in one unit.
Triplicate – using three copies (of hardware, software, messaging, power supplies, etc.). for redundancy and error control.
TRL – technology readiness level.
Trojan, - smaller body in same orbit as the primary, ahead or behind. For example, Earth has a Trojan.
TT&C – tracking, telemetry, and command.
TX - transmit
UART – Universal Asynchronous Receiver Transmitter.
UHF – ultra high frequency, 300 MHz to 3 GHz.
UNIX – operating system from Bell Labs.
uplink – from ground to space.
Upload – sending code to a computer, or data to a Cubesat in orbit. Or, on Mars. Maybe past Pluto.
USB – universal serial bus.
VHF – very high frequency – 30-300 MHz radio band.
Watchdog – hardware/software function to sanity check the hardware, software, and process; applies corrective action if a fault is detected; fail-safe mechanism.
WiFi – wireless networking.
X-band – 8-12 GHz radio band.
Yagi – a type of directional antenna, with multiple, parallel rods – like my old TV antenna.
Zombie-sat – a dead satellite, in orbit.

If you enjoyed this book, you might also be interested in some of these.

Stakem, Patrick H. *16-bit Microprocessors, History and Architecture*, 2013 PRRB Publishing, ISBN-1520210922.

Stakem, Patrick H. *4- and 8-bit Microprocessors, Architecture and History*, 2013, PRRB Publishing, ISBN-152021572X,

Stakem, Patrick H. *Apollo's Computers,* 2014, PRRB Publishing, ISBN-1520215800.

Stakem, Patrick H. *The Architecture and Applications of the ARM Microprocessors,* 2013, PRRB Publishing, ISBN-1520215843.

Stakem, Patrick H. *Earth Rovers: for Exploration and Environmental Monitoring,* 2014, PRRB Publishing, ISBN-152021586X.

Stakem, Patrick H. *Embedded Computer Systems, Volume 1, Introduction and Architecture*, 2013, PRRB Publishing, ISBN-1520215959.

Stakem, Patrick H. *The History of Spacecraft Computers from the V-2 to the Space Station*, 2013, PRRB Publishing, ISBN-1520216181.

Stakem, Patrick H. *Floating Point Computation*, 2013, PRRB Publishing, ISBN-152021619X.

Stakem, Patrick H. *Architecture of Massively Parallel Microprocessor Systems*, 2011, PRRB Publishing, ISBN-1520250061.

Stakem, Patrick H. *Multicore Computer Architecture,* 2014, PRRB

Publishing, ISBN-1520241372.

Stakem, Patrick H. *Personal Robots*, 2014, PRRB Publishing, ISBN-1520216254.

Stakem, Patrick H. *RISC Microprocessors, History and Overview,* 2013, PRRB Publishing, ISBN-1520216289.

Stakem, Patrick H. *Robots and Telerobots in Space Application*s, 2011, PRRB Publishing, ISBN-1520210361.

Stakem, Patrick H. *The Saturn Rocket and the Pegasus Missions, 1965,* 2013, PRRB Publishing, ISBN-1520209916.

Stakem, Patrick H. *Microprocessors in Space*, 2011, PRRB Publishing, ISBN-1520216343.

Stakem, Patrick H. Computer *Virtualization and the Cloud*, 2013, PRRB Publishing, ISBN-152021636X.

Stakem, Patrick H. *What's the Worst That Could Happen? Bad Assumptions, Ignorance, Failures and Screw-ups in Engineering Projects, 2014,* PRRB Publishing, ISBN-1520207166.

Stakem, Patrick H. *Computer Architecture & Programming of the Intel x86 Family, 2013,* PRRB Publishing, ISBN-1520263724.

Stakem, Patrick H. *The Hardware and Software Architecture of the Transputer*, 2011,PRRB Publishing, ISBN-152020681X.

Stakem, Patrick H. *Mainframes, Computing on Big Iron*, 2015, PRRB Publishing, ISBN- 1520216459.

Stakem, Patrick H. *Spacecraft Control Centers*, 2015, PRRB Publishing, ISBN-1520200617.

Stakem, Patrick H. *Embedded in Space,* 2015, PRRB Publishing,

ISBN-1520215916.

Stakem, Patrick H. *A Practitioner's Guide to RISC Microprocessor Architecture*, Wiley-Interscience, 1996, ISBN 0471130184.

Stakem, Patrick H. *Cubesat Engineeering*, PRRB Publishing, 2017, ISBN-1520754019.

Stakem, Patrick H. *Cubesat Operations*, PRRB Publishing, 2017, ISBN-152076717X.

Stakem, Patrick H. *Interplanetary Cubesats*, PRRB Publishing, 2017, ISBN-1520766173 .

Stakem, Patrick H. Cubesat Constellations, Clusters, and Swarms, Stakem, PRRB Publishing, 2017, ISBN-1520767544.

Stakem, Patrick H. *Graphics Processing Units, an overview*, 2017, PRRB Publishing, ISBN-1520879695.

Stakem, Patrick H. *Intel Embedded and the Arduino-101, 2017,* PRRB Publishing, ISBN-1520879296.

Stakem, Patrick H. *Orbital Debris, the problem and the mitigation*, 2018, PRRB Publishing, ISBN-*1980466483.*

Stakem, Patrick H. *Manufacturing in Space*, 2018, PRRB Publishing, ISBN-1977076041.

Stakem, Patrick H. , *NASA's Ships and Planes*, 2018, PRRB Publishing, ISBN-1977076823.

Stakem, Patrick H. *Space Tourism*, 2018, PRRB Publishing, ISBN-1977073506.

Stakem, Patrick H. *STEM – Data Storage and Communications*, 2018, PRRB Publishing, ISBN-1977073115.

Stakem, Patrick H. *In-Space Robotic Repair and Servicing*, 2018, PRRB Publishing, ISBN-1980478236.

Stakem, Patrick H. *Introducing Weather in the pre-K to 12 Curricula, A Resource Guide for Educators*, 2017, PRRB Publishing, ISBN-1980638241.

Stakem, Patrick H. *Introducing Astronomy in the pre-K to 12 Curricula, A Resource Guide for Educators*, 2017, PRRB Publishing, ISBN-198104065X.
Also available in a Brazilian Portguese edition, ISBN-1983106127.

Stakem, Patrick H. *Deep Space Gateways, the Moon and Beyond*, 2017, PRRB Publishing, ISBN-1973465701.

Stakem, Patrick H. *Crewed Spacecraft*, 2017, PRRB Publishing, ISBN-1549992406.

Stakem, Patrick H. *Rocketplanes to Space*, 2017, PRRB Publishing, ISBN-1549992589.

Stakem, Patrick H. *Crewed Space Stations,* 2017, PRRB Publishing, ISBN-1549992228.

Stakem, Patrick H. *Enviro-bots for STEM: Using Robotics in the pre-K to 12 Curricula, A Resource Guide for Educators,* 2017, PRRB Publishing, ISBN-1549656619.

Stakem, Patrick H. *STEM-Sat, Using Cubesats in the pre-K to 12 Curricula, A Resource Guide for Educators*, 2017, ISBN-1549656376.

Stakem, Patrick H. *Visiting the NASA Centers, and Locations of Historic Rockets and Spacecraft,* 2107, PRRB Publishing, ISBN-154965120X.

Stakem, Patrick H. *Lunar Orbital Platform-Gateway*, 2018, PRRB Publishing, ISBN-1980498628.

Stakem, Patrick H. Embedded GPU's, 2018, PRRB Publishing, ISBN- 1980476497.

Stakem, Patrick H. Mobile Cloud Robotics, 2018, PRRB Publishing, ISBN- 1980488088

Stakem, Patrick H. *Extreme Environment Embedded Systems,* 2017, PRRB Publishing, ISBN-1520215967.

Stakem, Patrick H. *What's the Worst, Volume-2*, 2018, ISBN-1981005579.

Stakem, Patrick H., *Spaceports*, 2018, ISBN-1981022287.

Stakem, Patrick H., *Space Launch Vehicles*, 2018, ISBN-1983071773.

Stakem, Patrick H. *Mars*, 2018, ISBN-1983116902.

Stakem, Patrick H. *X-86, 40th Anniversary ed*, 2018, ISBN-1983189405.

Stakem, Patrick H. *Lunar Orbital Platform-Gateway*, 2018, PRRB Publishing, ISBN-1980498628.

Stakem, Patrick H. *Space Weather*, 2018, ISBN-1723904023.

Stakem, Patrick H. *STEM-Engineering Process*, 2017, ISBN-1983196517.

2018 Release

Space Telescopes

Exoplanets

Planetary Defense

Exploration of the Asteroid Belt.